UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

2015-1427

HEWLETT-PACKARD CO.,

Appellant,

v.

MPHJ TECHNOLOGY INVESTMENTS,

Appellee.

On Appeal from the United States Patent and Trademark Office
Case No. IPR2013-00309

OPENING BRIEF OF APPELLANT HEWLETT-PACKARD CO.

CHARLENE M. MORROW
cmorrow@fenwick.com
STUART P. MEYER
smeyer@fenwick.com
RAVI RAGAVENDRA RANGANATH
rranganath@fenwick.com
MARION N. G. MILLER
mmiller@fenwick.com
FENWICK & WEST LLP
801 California Street
Mountain View, CA 94041
Telephone: (650) 988-8500
Facsimile: (650) 938-5200

Counsel for Appellant
HEWLETT-PACKARD CO.

Dated: May 11, 2015

CORPORATE DISCLOSURE STATEMENT

Pursuant to Federal Rules of Appellate Procedure 28 and 26.1, Appellant Hewlett-Packard Company states that no parent corporations and no publicly held companies own 10 percent or more of its stock.

CERTIFICATE OF INTEREST

Counsel for the Appellant Hewlett-Packard Company certifies:

1. The full name of every party or amicus represented by me is:

   Hewlett-Packard Company

2. The name of the real party in interest (if the party named in the caption is not the real party in interest) represented by me is:

   None.

3. All parent corporations and any publicly held companies that own 10 percent or more of the stock of the party or amicus curiae represented by me are:

   None.

4. The names of all law firms and the partners or associates that appeared for the party or amicus now represented by me in the trial court or agency or are expected to appear in this case are:

Law Firm	**Attorneys**
Fenwick & West LLP	Charlene M. Morrow Stuart P. Meyer Jennifer R. Bush Ravi Ragavendra Ranganath Marion N. G. Miller

Dated: May 11, 2015

By: *_/s/ Charlene Morrow_*
Charlene M. Morrow
Counsel for Appellant

TABLE OF CONTENTS

Page

CORPORATE DISCLOSURE STATEMENT i

CERTIFICATE OF INTEREST i

STATEMENT OF RELATED CASES 1

JURISDICTIONAL STATEMENT 1

STATEMENT OF ISSUES PRESENTED FOR REVIEW 2

STATEMENT OF THE CASE 3

SUMMARY OF THE ARGUMENT 11

ARGUMENT 15

I. STANDARD OF REVIEW 15

II. THE BOARD'S DECISION THAT SCANJET5 DOES NOT ANTICIPATE CLAIM 13 SHOULD BE REVERSED 15

III. THE BOARD IMPROPERLY BARRED REVIEW OF WHETHER CLAIM 13 IS OBVIOUS BY IMPROPERLY CONCLUDING THAT ANTICIPATION AND OBVIOUSNESS ARE REDUNDANT; THE AMERICA INVENTS ACT DOES NOT INSULATE THE BOARD FROM REVIEW OF SUCH CLEAR ERRORS 20

A. The Board Has Discretion to Act, But Cannot Act in a Manner Contrary to Law 20

B. Section 314 Does Not Bar Review of this Error by the Board 21

C. Anticipation and Obviousness Are Distinct Grounds for Invalidity 28

D. The Anticipation and Obviousness Defenses at Issue Here Are Not in Fact Redundant 29

TABLE OF CONTENTS
(continued)

Page

IV. THE BOARD'S AUTHORITY UNDER THE AMERICA INVENTS ACT DOES NOT INCLUDE DISCRETION TO REFUSE TO ADDRESS A PROPERLY PRESENTED GROUND FOR INVALIDITY 31

V. THE BOARD FAILED TO DISCHARGE ITS DUTIES UNDER THE ADMINISTRATIVE PROCEDURES ACT BY LEAVING UNRESOLVED ISSUES PRESENTED IN APPELLANT'S PETITION FOR *INTER PARTES* REVIEW...................... 40

A. *Inter Partes* Review Under the America Invents Act Is Subject to the Administrative Procedures Act's Standards for Formal Adjudication—Standards the Board Did Not Meet Here. 40

1. *Inter Partes* Review Has the Characteristics of Formal Adjudication. 40

2. *Inter Partes* Review Meets Tests for Formal Adjudication Under Supreme Court Law and the Law of Sister Circuits. 44

B. The Board Violated the Administrative Procedures Act's Formal Adjudication Requirements by Failing to Issue a Final Written Decision or Hold a Hearing on All Issues Presented in the Petition... 47

C. The Board's Failure to Address Grounds in Hewlett-Packard's Appeal Also Fails Under the More Lenient Informal Adjudication Standard.. 52

CONCLUSION AND STATEMENT OF RELIEF SOUGHT 56

CERTIFICATE OF SERVICE 57

CERTIFICATE OF COMPLIANCE.. 58

TABLE OF AUTHORITIES

Page(s)

CASES

Abbot Labs. v. Cordis Corp.,
710 F.3d 1318 (Fed. Cir. 2013) 41, 45

Abbott Labs. v. Gardner,
387 U.S. 136 (1967)....................27, 45

Amerijet Int'l, Inc. v. Pistole,
753 F.3d 1343 (D.C. Cir. 2014)....................52, 53, 54, 55

Argo-Coller Truck Lines v. United States,
611 F.2d 149 (6th Cir. 1979) 48

Bowen v. Mich. Acad. of Family Physicians,
476 U.S. 667 (1986)....................22, 27

Brand v. Miller,
487 F.3d 862 (Fed. Cir. 2007) 40

Butte Cnty., Cal. v. Hogen,
613 F.3d 190 (D.C. Cir. 2010)....................55

Citizens State Bank of Marshfield, Mo. v. FDIC,
718 F.2d 1440 (7th Cir. 1983) 49, 50

City of Gillette, Wyo. v. FERC,
737 F.2d 883 (10th Cir. 1984) 53, 54

Cohesive Techs, Inc. v. Waters Corp.,
543 F.3d 1351 (Fed. Cir. 2008) 28

Deckers Corp. v. United States,
752 F.3d 949 (Fed. Cir. 2014) 26

Dell, Inc., Hewlett-Packard Co., and NetApp, Inc. v. Elecs. & Telecomms. Research Inst.,
IPR2015-00549, Paper 10 (P.T.A.B. Mar. 26, 2015)....................39

TABLE OF AUTHORITIES
(continued)

Page(s)

Empire HealthChoice Assur., Inc. v. McVeigh,
547 U.S. 677 (2006)..38

Five Points Rd. Joint Venture v. Johanns,
542 F.3d 1121 (7th Cir. 2008)..45, 46

Friends of the Earth v. Reilly,
966 F.2d 690 (D.C. Cir. 1992)..46

Genentech, Inc. v. Chiron Corp.,
112 F. 3d 495 (Fed. Cir. 1997)..5

Google Inc. v. Jongerius Panoramic Techs., LLC,
IPR2013-00191, Paper No. 50 (P.T.A.B. Feb. 13, 2014)..................................43

Google v. B.E. Tech., LLC,
IPR2014-00031, Paper 9 (P.T.A.B. Apr. 9, 2014)..25, 38

Hewlett-Packard Co. v. Packard Press, Inc.,
281 F.3d 1262 (Fed. Cir. 2002)..20

In re Cuozzo Speed Techs., LLC,
479 F.3d 1271 (Fed. Cir. 2015)..23, 25, 26, 28

In re Dominion Dealer Solutions, LLC,
749 F.3d 1379 (Fed. Cir. 2014)..25

In re Gartside,
203 F.3d 1305 (Fed. Cir. 2000)..15, 41, 42

In re Procter & Gamble Co.,
749 F.3d 1376 (Fed. Cir. 2014)..26

Int'l Ass'n of Bridge, Structural and Ornamental Iron Workers, AFL-CIO, Local No. 111 v. NLRB,
792 F.2d 241 (D.C. Cir. 1986)..50, 51

TABLE OF AUTHORITIES
(continued)

Page(s)

K & I Transfer & Storage, Inc. v. NLRB,
805 F.2d 749 (7th Cir. 1986) 48, 49

KSR Int'l Co. v. Teleflex, Inc.,
550 U.S. 398 (2007) 29

Marathon Oil Co. v. EPA,
564 F.2d 1254 (9th Cir. 1997) 47

McNary v. Haitian Refugee Ctr., Inc.,
498 U.S. 479 (1991) 22

Motor Vehicle Mfrs. Ass'n of U.S., Inc. v. State Farm Mut. Auto. Ins. Co., 463 U.S. 29 (1983) 52

On-Line Careline, Inc. v. Am. Online, Inc.,
229 F.3d 1080 (Fed. Cir. 2000) 20

Rexnord Indus., LLC v. Kappos,
705 F. 3d 1347 (Fed. Cir. 2013) 21

ScentAir v. Prolitec,
IPR2013-00179, Paper 9 (P.T.A.B. Apr. 16, 2013) 43

Se. Cmty. Coll. v. Davis,
442 U.S. 397 (1979) 27

Seacoast Anti-Pollution League v. Costle,
572 F.2d 872 (1st Cir. 1978), *superseded by the law of the circuit rule, as recognized in Dominion Energy Brayton Point, LLC v. Johnson*, 443 F.3d 12 (1st Cir. 2006) 46

St. Jude Med., Cardiology Div., Inc. v. Volcano Corp.,
749 F.3d 1373 (Fed. Cir. 2014) 26

Steadman v. SEC,
450 U.S. 91 (1981) 44, 45

TABLE OF AUTHORITIES
(continued)

Page(s)

Teamsters v. Daniel,
439 U.S. 551 (1979).....27

Tourus Records Inc. v. DEA,
259 F.3d 731 (D.C. Cir. 2001).....53, 54

Volkswagen Grp. of Am. v. Farlight LLC,
IPR2013-00238, Paper 15 (P.T.A.B. Sept. 26, 2013) 25, 38

Zeiss v. Nikon,
IPR2013-00362, Paper 14 (P.T.A.B. Jan. 13, 2014) 44

STATUTES AND RULES

9 V.S.A. § 4197(a) 6

37 C.F.R. § 42.104(b)(2).....31

37 C.F.R. § 90.3 1

77 Fed. Reg. 48,756 (Aug. 14, 2012) (codified at 37 C.F.R. pt. 42).....31, 35

5 U.S.C. § 551(5) 41

5 U.S.C. § 551(6) 41

5 U.S.C. § 555(e) 52, 53, 54, 55

5 U.S.C. § 556.....40, 47, 51

5 U.S.C. § 556(b) 47

5 U.S.C. § 556(c) 47

5 U.S.C. § 557.....40, 47, 52, 53

5 U.S.C. § 557(c) .....*passim*

5 U.S.C. § 557(c)(3)(A) 49

TABLE OF AUTHORITIES
(continued)

Page(s)

5 U.S.C. § 704 22

5 U.S.C. § 706(2) 21

5 U.S.C. § 706(2)(C) 32, 35

28 U.S.C. § 1295(4)(A) 1, 42

35 U.S.C. § 101 38

35 U.S.C. § 102 .... 28, 31, 38

35 U.S.C. § 103 .... 28, 31, 38

35 U.S.C. § 112 38

35 U.S.C. § 251(d) 38

35 U.S.C. § 307 32

35 U.S.C. § 311 .... 1, 23, 24

35 U.S.C. § 311(a) 23

35 U.S.C. § 311(b) 29, 38

35 U.S.C. § 312 24

35 U.S.C. § 312(d) 24

35 U.S.C. § 313 23

35 U.S.C. § 314 .... *passim*

35 U.S.C. § 314(a) 23, 24, 27, 31

35 U.S.C. § 314(b) 23

35 U.S.C. § 314(d) .... *passim*

TABLE OF AUTHORITIES
(continued)

Page(s)

35 U.S.C. § 315 24

35 U.S.C. § 315(a) 26

35 U.S.C. § 315(b) 26

35 U.S.C. § 315(e) 23

35 U.S.C. § 315(e)(1) 39

35 U.S.C. § 316 .... 9, 23, 24, 41

35 U.S.C. § 316(a)(1) 41, 45

35 U.S.C. § 316(a)(5) 45

35 U.S.C. § 316(a)(6) 45

35 U.S.C. § 316(a)(10) 41, 45

35 U.S.C. § 316(b) 34, 35

35 U.S.C. § 317(a) 23

35 U.S.C. § 318 .... 24, 41

35 U.S.C. § 318(a) 23, 32, 33

35 U.S.C. § 318(c) 23

35 U.S.C. § 319 .... 1, 42, 45

35 U.S.C. § 329 1

Administrative Procedures Act, Pub. L. 79-404, 60 Stat. 237 (1946) .... *passim*

The America Invents Act, Pub. L. 112-29, 125 Stat. 325 (2011) .... *passim*

Federal Rule of Evidence 201(b)(2) 5

TABLE OF AUTHORITIES
(continued)

Page(s)

OTHER AUTHORITIES

153 Cong. Rec. H10280 (Sept. 7, 2007) 33

154 Cong. Rec. S9989 (daily ed. Sept. 27, 2008) 33

157 Cong. Rec. S1111 (Mar. 2, 2011) 33, 43

157 Cong. Rec. S1376 (daily ed. Mar. 8, 2011) 33, 34

157 Cong. Rec. S1377 (daily ed. Mar. 8, 2011) 35, 36

110 S. Rep. No. 110-259 (2008) 33

H.R. Rep. No. 112-98, pt. 1 (2011) 33, 34, 41, 43

Patent Reform Act of 2005, H.R. 2795, 109th Cong. (2005) 38

STATEMENT OF RELATED CASES

State of Vt. v. MPHJ Tech. Invs. LLC, No. 15-1310 (Fed. Cir.), a proceeding about MPHJ Technology Investments, LLC's assertion of U.S. Patent No. 6,771,381, is pending. For the second time, MPHJ appeals the U.S. District Court for the District of Vermont's remand to state court. The State of Vermont filed a complaint in state court against MPHJ under the Vermont Consumer Protection Act. *State of Vt. v. MPHJ Tech. Invs., LLC*, No. 282-5-13, Dkt. No. 1 (Vt. Super. Ct. May 8, 2013). MPHJ removed the case to federal court, and the U.S. District Court remanded the case to state court because the State's original complaint did not create federal jurisdiction. *State of Vt. v. MPHJ Tech. Invs., LLC*, No. 2:13-cv-00170, Dkt. No. 1 (D. Vt. June 7, 2013); *State of Vt. v. MPHJ Tech. Invs., LLC*, No. 2:13-cv-00170, Dkt. No. 61 (D. Vt. Apr. 15, 2014). MPHJ's appeal of that decision to this Court was dismissed. *State of Vt. v. MPHJ Tech. Invs., LLC*, No. 14-1481, Dkt. No. 24 (Fed. Cir. Aug. 11, 2014). On remand, the State filed an Amended Complaint. *State of Vt. v. MPHJ Tech. Invs., LLC*, No. 282-5-13 (Vt. Super. Ct. Aug. 28, 2014). MPHJ removed to federal court again, based on the amended complaint. *State of Vt. v. MPHJ Tech. Invs., LLC*, No. 2:14-cv-00192, Dkt. No. 1 (D. Vt. Sept. 9, 2014). The U.S. District Court again remanded to state court. *State of Vt. v. MPHJ Tech. Invs., LLC,* No. 2:14-cv-00192, Dkt. No. 33 (D. Vt. Jan 12, 2015). MPHJ now appeals that second remand.

JURISDICTIONAL STATEMENT

The Patent Trial and Appeal Board has jurisdiction under 35 U.S.C. Sections 311–319 over petitions for *inter partes* review of a United States patent. This Court has jurisdiction under 28 U.S.C. Section 1295(4)(A) over appeals taken pursuant to 35 U.S.C. Section 319 from decisions of the Patent Trial and Appeal Board with respect to petitions for *inter partes* review. Hewlett-Packard Company initiated *Inter Partes* Review No. IPR2013-00309. A final decision of the Patent and Trial Appeal Board was issued on November 19, 2014. A42. This appeal, filed on January 21, 2015, is timely under 35 U.S.C. Section 329 and 37 C.F.R. Section 90.3.

STATEMENT OF ISSUES PRESENTED FOR REVIEW

1. Whether the Board improperly concluded that claim 13 of U.S. Patent No. 6,771,381 is not anticipated by ScanJet5.

2. Whether the Board's conclusion that obviousness was cumulative of anticipation was legally erroneous.

3. Whether the Board's refusal to review each of the grounds presented to it was an improper exercise of discretion not authorized by the America Invents Act.

4. Whether the Board's refusal to substantively address each of the grounds presented to it was in violation of the Administrative Procedures Act, which requires the agency to act on each ground presented to it.

STATEMENT OF THE CASE

Because the patent holder, MPHJ Technology Investments, LLC ("MPHJ"), was asserting the patent in suit against Hewlett-Packard Company ("Hewlett-Packard") customers, Hewlett-Packard filed the proceedings below. A96. Hewlett-Packard sought a ruling that MPHJ's patent claims were invalid because they were anticipated or made obvious by Hewlett-Packard's own prior art printing solutions. Hewlett-Packard obtained a ruling that all but one of the claims of U.S. Patent No. 6,771,381 ("'381 patent") is invalid over Hewlett-Packard prior art publications. A87.

Unfortunately, the Board refused to invalidate claim 13 of the '381 patent. Hewlett-Packard submits that this decision was in error for two reasons: first, the Board should have concluded based on the record before it that claim 13 was anticipated; and second, the Board should have undertaken review of whether claim 13 was obvious over the cited art and invalidated on that ground.

The patent at issue is held by MPHJ. Following its acquisition of the patent rights, MPHJ embarked on a campaign of sending letters to 16,465 small businesses, demanding each pay $1,000 or more per employee to use the patented technology. Up to three letters were sent to each business, with an increasingly aggressive tone. The letters alleged that "[a] good example of an infringing system, and one your company likely uses, is an office local area network ('LAN') which is in communication with a server, employee computers having email software such as Outlook or Lotus, and a third-party scanner (or a multi-function printer with scanning functionality) which permits the scanning of a document directly to employee email address as a pdf attachment. Such a system would be a typical example of what infringes." A2229. The letters also alleged that MPHJ "ha[s] had a positive response from the business community" and that most businesses "are interested in operating lawfully and [have] tak[en] a license promptly." A2231. The third letter stated that non-response by the addressee was assumed to mean "that the system you are using is covered by the patents. In that case, you do need a license." A2249. The third letter also enclosed a sample complaint and threatened that, if no reply was received within two weeks, the complaint would be filed in federal district court. *Id*.

The Federal Trade Commission filed a complaint against MPHJ last year about this licensing campaign and this year announced a settlement with MPHJ over its "deceptive acts" and phony legal threats. *See In the Matter of MPHJ Tech. Invs., LLC*, *et al.*, No. 142 3003, Dkt. No. C-4513 (F.T.C. Nov. 6, 2014). The settlement calls for MPHJ to stop making deceptive claims in their infringement assertion letters, such as falsely stating that other businesses had already paid for licenses, or threatening to file suits it had no intention of actually initiating. In particular, the settlement prohibits MPHJ from "[m]ak[ing] any representation in a

Patent Assertion Communication, expressly or by implication, that [MPHJ] will take any action with respect to the filing of a Lawsuit" without "competent and reliable evidence sufficient to substantiate that they are prepared to and able to take the action necessary to make the representation true." *See id.*, Agreement Containing Consent Order, at 4–5. The State of Vermont similarly filed a consumer protection complaint against MPHJ, alleging unfair and deceptive trade practices. *State of Vt. v. MPHJ Tech. Invs., LLC*, No. 282-5-13, Dkt. No. 1 (Vt. Super. Ct. May 8, 2013).[1] In addition to the State of Vermont, at least three other states have also investigated MPHJ's conduct. *State of Minn. v. MPHJ Tech. Invs., LLC*, No. 62-cv-13-6080 (Minn. 2d Dist.); *In the Matter of MPHJ Tech. Invs., LLC*, No. 14-015 (N.Y. Atty. Gen.); *Activision TV, Inc. v. Pinnacle Bancorp., et al.*, No. 8:13-cv-00215 (D. Neb.). Minnesota and New York entered into agreements with MPHJ to limit MPHJ's licensing campaign. *See* Assurance of Discontinuance, *State of Minn. v. MPHJ Tech. Invs.*, LLC, No. 62-cv-13-6080 (Minn. Dist. Ct. Aug. 20, 2013); Assurance of Discontinuance, *In the Matter of the Investigation of MPHJ Tech. Invs.*, No. 14-015 (N.Y. Att'y Gen. Jan. 13, 2014)

[1] The Court may take judicial notice of the records of official government proceedings. FED. R. EVID. 201(b)(2); *Genentech, Inc. v. Chiron Corp.*, 112 F. 3d 495, 497 & n.1 (Fed. Cir. 1997) (taking judicial notice of papers filed before the Board of Patent Appeals and Interferences because "the interference record before the Board is a public record, . . . capable of accurate and ready determination by resort to unquestionable sources").

MPHJ's actions, along with those of other non-practicing entities who were overly aggressive in assertion of the their patent rights also led to statutory reform in the State of Vermont, in the form of Vermont's "Bad Faith Assertions of Patent Infringement" law as Chapter 120 to Title 9 of Vermont's Statutes. The statutes expressly provide that the bad faith assertion of patent infringement is prohibited. 9 V.S.A. § 4197(a). The deterrents contained in these provisions are substantial, including a requirement that an entity asserting patent infringement post a sizeable prosecution bond (up to $250,000) in the event the court determines that the entity's actions are taken in bad faith. Twenty-three other states have also enacted, or are in the process of enacting, laws prohibiting bad-faith demand letters.[2] Federal legislation has also been proposed, but not adopted, that would bar sending

[2] Ala. S.B. 121 § 2(d)(1)-(2) (Act No. 2014-218); Ga. H.B. 809, Sec. 1, § 10-1-770(b)(1)-(2) (Act. No. 513); Idaho S.B. 1354, Sec. 1, § 48-1703(2)(a)-(c) (Session Law Ch. 277); Ill. S.B. 3405, Sec. 5, § 2RRR(b)(4)(C) (Public Act No. 098-1119); La. S.B. 255 Sec. 1, § 1428(B)(2)(a)-(b) (Act. No. 297); Me. S.P. 654, Sec. 1, § 8701(3)(A)(1) and (3) (Public Law Ch. 543); Md. S.B. 585, Sec. 1, § 11-1603(B)(I) and (III) (Ch. 307); Mich. H.B. 5701 (introduced July 16, 2014); Mo. H.B. 1374 Sec. A, § 416.652(2)(1)(c) (signed by H. Speaker on May 6, 2015); N.H. S.B. 303, Ch. 359-M (II)(a)-(b) (Ch. 2014-0197); N.C. H.B. 1032, Sec. 1, § 75-139(a)(1)-(2) (Ch. Sess. Law 2014-110); Ohio H.B. 573 (referred to H. Judiciary Comm. on June 3, 2014); Okla. H.B. 2837, Sec. 2(A)(3)(d)(3) (Ch. 305); Or. S.B. 1540, Sec. 2(4)(b) and (d) (Ch. 19, 2014 Laws); Pa. S.B. 1222 (referred to H. Judiciary Comm. on Feb. 4, 2014); R.I. SB 2822 (referred to H. Judiciary Comm. on June 19, 2014); S.C. H.B. 4629 (introduced Feb. 6, 2014); S.D. S.B. 143, Sec. 3(1)-(2) (signed by Governor on Mar. 26, 2014); Tenn. H.B. 2117, Sec.1, § 29-40-102(b)(1)-(2) (Pub. Ch. 879); Tex. S.B. No. 1457 (referred to Judiciary & Civil Jurisprudence Comm. on May 5, 2015); Utah H.B. 117, Sec. 1, § 78b-6-1903(2)(a)(v) (Ch. No. 310); Va. H.B. 375, Ch.18.1, § 59.1-215.2(B)(1)-(2) (Ch. No. 10); Wis. Act No. 339, Sec. 1, § 100.197(2)(a).

abusive demand letters. Protecting American Talent and Entrepreneurship Act ("PATENT Act"), S. 1137, at 32, 114th Cong. (2015).

In parallel, because MPHJ was sending its licensing letters to users of Hewlett-Packard multi-function printers, Hewlett-Packard filed the proceedings below, seeking a ruling that MPHJ's patent claims were invalid because they were anticipated or made obvious by Hewlett-Packard prior art printing solutions. A2266, A2272, A2279, A2285, A2292, A2299, A2306, A2314, A2318. Hewlett-Packard obtained a ruling that all but one of the claims of the '381 patent are invalid. A87.

Unfortunately, the Board refused to invalidate claim 13 of the '381 patent. Hewlett-Packard submits that this decision was in error for two reasons: first, the Board should have concluded based on the record before it that claim 13 was anticipated; and second, the Board should have undertaken review of whether claim 13 was obvious over the cited art and invalidated on that ground.

The Hewlett-Packard prior art at issue in the Board's final decision on anticipation is the Hewlett-Packard Network ScanJet 5 Scanner User's Guide (A1426–565). The only element the Board found missing—a list of available modules—is in fact taught in the ScanJet 5 User's Guide. A1357 (¶ 132) (ScanJet 5 uses "lists of destinations, workflows and the like [that are] started automatically when Windows is started"); *see* A1458–59; *see also* A38 ("SJ5 discloses

Windows-based modules"). The Board simply erred in its analysis by referring to evidence supporting the wrong claims, and thus did not consider the relevant section of the manual. In explaining its determination that Hewlett-Packard "[did] not show by a preponderance of evidence that [the ScanJet 5 User's Guide] anticipates claim 13," the Board stated that "[t]he Petition generally refers to the analysis of claims 1, 2, and 7[] to address the limitations in claim 13," even though "those claims recite different elements." A87 (citing A2277). But in its Petition, on the very page the Board cites to support its final decision, Hewlett-Packard also refers to its analysis of claim 9 to address the limitations in claim 13. A2277; *see* A2275–76. The Board's final decision thus did not take into account Hewlett-Packard's underlying analysis of the reasons the ScanJet 5 User's Guide anticipates claim 9. Had the Board conducted a review of all of the evidence submitted to it, it should have concluded that claim 13 is anticipated.

Had the Board also undertaken review of the obviousness ground presented in the petition, the Hewlett-Packard prior art that would have been at issue is a Press Release related to the ScanJet 5, entitled "HP Introduces Next-Generation Network Scanner." A2318–23. It contains additional detail on how the Hewlett-Packard scanners interface with the Windows operating system. A2215 ("support for the Network ScanJet 5 scanner has been expanded to include Microsoft® Windows® NT" along with "Windows 3.1[and] Windows 95"), A2217 (the "scanner supports . . . Microsoft Windows NT Server 3.51 or 4.0; and . . . networks running Windows 3.1x, Windows 95 and Windows NT 3.51 client PCs"). However, the Board refused to address this ground presented in the Petition, instead stating without further explanation that this Hewlett-Packard obviousness ground was "denied as redundant in light of the determination that there is a reasonable likelihood that the challenged claims are unpatentable based on the grounds of unpatentability on which we institute an *inter partes* review." A38.

As a result, under 35 U.S.C. Section 316, all but one claim of the '381 patent will be cancelled. However, given the partial disposition of this matter, as a result of the incomplete review by the Board of the prior art presented to it, Hewlett-Packard has not been able to obtain complete relief for its customers. While MPHJ should be deterred from asserting the remaining claim given the clear evidence of obviousness submitted by Hewlett-Packard, the PTO's cursory and legally erroneous rejection of this combination of art as "redundant" casts that into doubt. For example, in its First Amended Complaint against the State of Vermont, MPHJ states that it "wishes to send letters inquiring about infringement of its patents related to claims whose validity was recently upheld . . . following an *Inter Partes* Review." First Amended Complaint ¶ 6, *MPHJ Tech. Invs., LLC v. Sorrell*, No. 2:14-cv-00191, Dkt. No. 18 (D. Vt. Dec. 29, 2014) ("MPHJ Vt. Compl."). These "post-IPR letters" "make express reference to the results of the IPR." *Id.* In support of its alleged First Amendment Right to send these letters, MPHJ states that "[a]t least one claim of the . . . '381 Patent . . . is not invalid under Title 35 of the U.S. Code," and claims that "[o]n information and belief, at least one of the claims of the . . . '381 Patent . . . is infringed by at least one or more of the businesses in Vermont who received patent enforcement correspondence from MPHJ that is the subject of the First Amended Complaint." MPHJ Vt. Compl. ¶¶ 92*,* 97. Addressing another of its patents for which a single claim recently survived *inter partes* review, MPHJ notes that it "sensibly desires now to address the widespread infringement it reasonably suspects is occurring in Vermont." *Id.* ¶ 119. Accordingly, Hewlett-Packard and its customers have suffered a concrete harm as a result of the Board's actions.

SUMMARY OF THE ARGUMENT

The Board erred by reviewing only a portion of the teaching in the ScanJet5 reference. Had it undertaken review of the passages from the prior art reference to which it was directed by Hewlett-Packard's petition, it would not have been able to conclude that one of the claim limitations of claim 13 was not taught. The Hewlett-Packard prior art at issue in the Board's final decision on anticipation is the Hewlett-Packard Network ScanJet 5 Scanner User's Guide. A1426–565. The only element the Board found missing—a list of available modules—is in fact taught in the reference. A1357 (¶ 132) (ScanJet 5 uses "lists of destinations, workflows and the like, [that are] started automatically when Windows is started"); *see* A1458–59; *see also* A38 ("SJ5 discloses Windows-based modules"). The Board simply erred in its analysis by referring to the wrong claims, and thus did not consider the relevant section of the manual. Explaining its determination that Hewlett-Packard "[did] not show by a preponderance of evidence that [the ScanJet 5 User's Guide] anticipates claim 13," the Board stated that "[t]he Petition generally refers to the analysis of claims 1, 2, and 7[] to address the limitations in claim 13," even though "those claims recite different elements." A87 (citing A2277). But in its Petition, on the very page the Board cites to support its final decision, Hewlett-Packard also refers to its analysis of claim 9 to address the limitations in claim 13. A2277; *see* A2275–76. The Board's final decision thus did not take into account Hewlett-Packard's underlying analysis of the reasons the ScanJet 5 User's Guide anticipates claim 9. Had it conducted a review of all of the evidence submitted to it, the Board would have reached the conclusion that claim 13 is anticipated.

Had the Board also undertaken review of the obviousness ground presented in the petition, the Hewlett-Packard prior art that would have been at issue is a Press Release related to the ScanJet 5, entitled "HP Introduces Next-Generation Network Scanner." A2318–23. It contains additional detail on how the Hewlett-Packard scanners interfaced with the Windows operating system. A2215 ("support for the Network ScanJet 5 scanner has been expanded to include Microsoft® Windows® NT. . . [along with] Windows 3.1 [and] Windows 95"), A2217 (the "scanner supports . . . Microsoft Windows NT Server 3.51 or 4.0; and . . . networks running Windows 3.1x, Windows 95 and Windows NT 3.51 client PCs"). Had the Board undertaken the requested review of this combination, it should have found claim 13 invalid for obviousness over the cited Hewlett-Packard art.

The Board's refusal to review the obviousness ground on the basis that it was "redundant" represents legal error. This Court has been clear that anticipation and obviousness are separate legal grounds, and there is simply no legal basis for the Board's decision that an obviousness ground could be redundant in light of an anticipate ground. Nor could the Board's analysis be sustained on review of the record, as the references are not in fact redundant, and the analysis of obviousness distinct from that of anticipation.

The Board's refusal to review the obviousness ground presented to it was also an abdication of its authority under the America Invents Act ("AIA"). The AIA does not give the Board the discretion to refuse to act on part of an otherwise meritorious petition. Nor can the Board shield its abuse of discretion from review by relying on the narrow exception to appellate review provided in 35 U.S.C. Section 314(d), which only bars review of *the threshold decision whether to institute* and not of its conduct in a proceeding once it has found a basis for initiation. In any event, even if section 314(d) applied, the Board's decision is still reviewable for abuse and for compliance with its obligations under the Administrative Procedures Act ("APA"). Pub. L. 79-404, 60 Stat. 237 (1946).

The Board's refusal to review the obviousness ground presented to it was also legally erroneous because under the APA, when an agency is given the authority to act on the record, that agency has the obligation to address each issue fairly presented to it. *Id.* The AIA establishes such a hearing on the record, and accordingly, this matter should be remanded to the Board with instruction to adhere to the procedures mandated by the APA. Pub. L. 112-29, 125 Stat. 325 (2011).

Finally, the Board's refusal to substantively address each of the grounds presented to it is a violation of the APA that requires remand for further decision-making by the Board.

Accordingly, this matter should be remanded to the Board for further proceedings.

ARGUMENT

I. STANDARD OF REVIEW

The factual basis for the Board's final decision on anticipation is reviewed under a "substantial evidence" standard. *See In re Gartside*, 203 F.3d 1305, 1312–1316 (Fed. Cir. 2000). The Board's conduct is reviewed *de novo* where legal error is at issue. *Id.*

II. THE BOARD'S DECISION THAT SCANJET5 DOES NOT ANTICIPATE CLAIM 13 SHOULD BE REVERSED

The Board's decision that ScanJet5 does not anticipate claim 13 of the '381 patent because it does not teach the recited "list of available module means" is not supported by substantial evidence. The Board concluded that Hewlett-Packard "fails, among other things, to specify the required lists of modules." A87. This Board decision is reversible because it is "unsupported by substantial evidence." *Gartside*, 203 F.3d 1305, 1311–1312, 1327.

Claim 13 recites a "list of available module means." A245 (87:50). The '381 patent teaches deployment of the invention as a "standard COM component," and incorporates by reference Windows teachings from Microsoft. A228 (53:30-48). The '381 patent's preferred embodiment teaches that to "Maintain [a] List of Available Modules," the Windows registry may be used. The patent explains that the "Windows registry contains the list of available Input, Output, and Process Modules that can be used with VC." A238 (74:13-15, 7:54-55); *see also* A189

(FIG. 36 and associated text). This claimed "list of available module means" is thus provided by using the Windows registry infrastructure that is, as the specification admits, background technology in "specifications designed by Microsoft, published in the technical literature, and incorporated herein by reference." A228 (53:30-33).

The corresponding structure in the Hewlett-Packard ScanJet5 prior art is the ScanJet utility. It uses "lists of destinations, workflows and the like, [that are] started automatically when Windows is started." A1357 (¶ 132); *see* A1458–59. The ScanJet utility can use these lists to maintain "lists of destinations, addresses, workflows, etc.," configurable by a user. A1357 (¶ 133); s*ee* A1465.

The Board concluded in the Institution Decision that the "list of available module means" corresponds to a generic listing algorithm and memory to store, or point to, a list of modules. A19. This construction was never traversed by MPHJ.

In its Final Decision analyzing claim 13, the Board incorrectly restricted itself to the evidence proffered in support of anticipation of claims 1, 2, and 7. A87 (stating that the Petition's analysis of claim 13 "generally refers to the analysis of claims 1, 2, and 7, to address the limitations in claim 13."). The Board then concluded that the evidence proffered in connection with claims 1, 2, and 7 was not probative as "those claims recite different elements." *Id.* Had the Board reviewed all of the evidence cited by Hewlett-Packard, it would have reached a different conclusion.

The Board also should have reviewed the evidence provided in support of anticipation of claim 9. A2277–78 (for each of the elements of claim 13, [13.P] to [13.4], Hewlett-Packard cited analysis of the corresponding element of claim 9, [9.P] to [9.4]). Claim 9 recites elements nearly identical to the elements of claim 13, except for their respective claim preambles, in which claim 9 is dependent on claim 7 and claim 13 is an independent claim.

Hewlett-Packard provided the ScanJet5 reference, and a 12- paragraph expert analysis of that evidence and comparing it to requirements in claims 9 and 13. A1356–58 (¶¶ 130–134); A1361 (¶ 146); *see* A1352–53 (¶¶ 118, 119); A1355 (¶¶ 126–128). This was not the same evidence cited on claims 1, 2, and 7. *Compare* 118, 119 (in the context of claim 1, stating that the ScanJet 5 reference "disclos[es] an imaging device used to capture images from paper using a scanner," "[s]elect[s] the scanner as an input source," and "discloses OCR software that manages electronic paper," and stating that"one skilled in the art would understand that a system that links to external applications would include a module capable of combining the digital imaging device with the linked external applications"); ¶ 120 (in the context of claim 2, stating that "ScanJet 5's external devices and applications include a printer, a fax machine, and a scanner"); *and*

¶ 127 (in the context of claim 7, stating that "users can create automatic workflows that are 'associated with a named group of settings'") *with* ¶ 132 (in the context of claim 9, stating further that "the ScanJet 5 Utility maintains lists of destinations, workflows and the like, and is started automatically when Windows is started" and that "[o]ne skilled in the art in 1998 would recognize from this description that a list of active processes and programs would be stored in a registry of the system").

Had the Board reviewed this evidence, it should have reached a different conclusion. The evidence of record discusses the disclosure of input, output, and process modules in ScanJet5. *See id.* (citing A1357–58 (¶¶ 132–133)) (discussing A1437, A1443, A1446, A1458–59, A1472, A1518, A1539). It explains that when Windows is started up, ScanJet5 maintains "destination lists" such as "workflows, faxes, or distribution lists" that are "started automatically when Windows is started." A1458–59. The ScanJet utility "allows a user to set up lists of destinations, addresses, workflows, etc., and such lists can be viewed 'at any time' on a profile tab." *See* A1357–58 (¶¶ 132–133) (citing A1458–59, A1465). This is, of course, a list of available modules.

In addition, Hewlett-Packard called out element [9.2], which is identical to element [13.2], as an area of strength for the ScanJet5 reference. *See generally,* A1420–24 (¶¶ 364–371). Hewlett-Packard explained that:

> [ScanJet 5] is a 1997 publication that most clearly discloses the claimed features of the '381 [patent]. For example, SJ5 discloses a

> scanner product that allows distribution to email addresses that expressly include Internet email addresses. In addition, SJ5 expressly discloses MAPI, a messaging API, which I discuss above in connection with interpretation of [9.2], and a 'Go' button, described in conjunction with [5.1].

A1421 (¶ 365).

Hewlett-Packard further discussed how the ScanJet5 Utilities can be used to fix missing email links and noted that one skilled in the art in 1998 would recognize from this description that to perform this function, a list of active processes and programs would be stored in a registry of the system. A1357 (¶ 132) (citing A1458–59, A1539). Specifically, since the Board held that "each "module," as recited in the claims, is a logically separable part of the claimed data management system" (A14–15), such a list of active processes and programs performing certain functions also meets the requirement of a list of available modules.

Indeed, the Board initially concluded in the Institution Decision that claim 13's "'maintain list of available module means' corresponds to a generic listing algorithm and memory that stores, or points to, a list of modules." A19. The Board thus concluded that since "SJ5 discloses Windows-based modules and a system that is similar to the disclosed invention, Petitioner establishes a reasonable likelihood of prevailing on the ground that [ScanJet 5] anticipates claims 1-15." A38 (citing A103–10). While that decision was made under the lower standard of proof for an institution decision, the Board should have reviewed the evidence to which it had been directed. On the basis of that evidence, the Board should have concluded that the record supported the same conclusion under evidentiary standard applicable to the final decision.

In *Hewlett-Packard Co. v. Packard Press, Inc*., this Court held that "[t]he substantial evidence standard requires that this court ask whether a reasonable person might accept that the evidentiary record adequately supports the Board's conclusion." 281 F.3d 1262, 1265 (Fed. Cir. 2002) (citing *On-Line Careline, Inc. v. Am. Online, Inc.,* 229 F.3d 1080, 1085 (Fed. Cir. 2000)). Here, the Board's final conclusion is contrary to the proffered record. Accordingly, this Court should reverse the Board decision that claim 13 is not anticipated and remand for further proceedings.

III. THE BOARD IMPROPERLY BARRED REVIEW OF WHETHER CLAIM 13 IS OBVIOUS BY IMPROPERLY CONCLUDING THAT ANTICIPATION AND OBVIOUSNESS ARE REDUNDANT; THE AMERICA INVENTS ACT DOES NOT INSULATE THE BOARD FROM REVIEW OF SUCH CLEAR ERRORS

A. The Board Has Discretion to Act, But Cannot Act in a Manner Contrary to Law

The bounds of the Board's power are underscored by the duty of the reviewing court to:

> [H]old unlawful and set aside agency action, findings, and conclusions found to be (A) arbitrary, capricious, an abuse of discretion, or otherwise not in accordance with law; (B) contrary to

> constitutional right, power, privilege, or immunity; (C) in excess of statutory jurisdiction, authority, or limitations, or short of statutory right; (D) without observance of procedure required by law; (E) unsupported by substantial evidence in a case subject to sections 556 and 557 of this title or otherwise reviewed on the record of an agency hearing provided by statute; or (F) unwarranted by the facts to the extent that the facts are subject to trial de novo by the reviewing court.

5 U.S.C. § 706(2). Here, the Board decision that an obviousness challenge was cumulative of an anticipation challenge was not in accordance with law. Accordingly, the Court should set aside that aspect of the Board's finding. *See Rexnord Indus., LLC v. Kappos*, 705 F. 3d 1347, 1355 (Fed. Cir. 2013) (reversing determination of nonobviousness where the Board "erred . . . in reversing the examiner's ruling of obviousness on [one] ground, without considering any of the other grounds of obviousness that had been raised for reexamination"). On remand, the Board should consider the obviousness challenge on its own merits.

B. Section 314 Does Not Bar Review of this Error by the Board

In other matters, the PTO has taken the position that section 314 bars review of a broad range of asserted errors in its administration of the America Invents Act ("AIA"). *See, e.g.,* Response of Intervenor – Director of The United States Patent and Trademark Office Opposing Hearing En Banc at 8, *In re Cuozzo Speed Techs., LLC*, No. 14-1301, Dkt. No. 68 (Fed. Cir. Apr. 14, 2015) (in opposition to request for rehearing en banc, asserting that "[section] 314(d) does not bar only challenges to the agency's determination under [section] 314(a) that a petitioner has not established a 'reasonable likelihood' of success on the merits"). However, the PTO's position is contrary to administrative decisional law.

A dominating consideration in review of agency actions is the "strong presumption that Congress intends judicial review of administrative action." *Bowen v. Mich. Acad. of Family Physicians*, 476 U.S. 667, 670 (1986). This presumption is embodied in the Administrative Procedures Act ("APA"), which provides that "final agency action for which there is no other adequate remedy in a court [is] subject to judicial review." 5 U.S.C. § 704; *see also* 5 U.S.C. § 702. In keeping with this presumption, 35 U.S.C. Section 314(d), which bars the review of "[t]he determination by the Director whether to institute an *inter partes* review *under this section* shall be final and nonappealable," should be read narrowly. 35 U.S.C. § 314(d) (emphasis added).

Section 314(d) only bars review of the threshold decision by the Board of whether the prior art warrants instituting *inter partes review.* It does not otherwise insulate the Board from the obligation to follow the law in discharging its duties. *See Bowen*, 476 U.S. at 678 (holding that "it is implausible to think [Congress] intended that there be *no* forum to adjudicate statutory and constitutional challenges to regulations promulgated") (emphasis in original); *see also McNary v. Haitian Refugee Ctr., Inc.*, 498 U.S. 479 (1991) (statute precluding direct review of decisions of Immigration and Naturalization Service ("INS") denying applications for Special Agricultural Worker status did not deprive district court of jurisdiction to consider due process challenge to manner in which those provisions were being administered by the INS); *In re Cuozzo Speed Techs., LLC*, 479 F.3d 1271, 1277 (Fed. Cir. 2015) (acknowledging that "mandamus may be available to challenge the PTO's decision to grant a petition to institute IPR after the Board's final decision in situations where the PTO has clearly and indisputably exceeded its authority").

In addition, section 314 only bars review of a very narrow aspect of the Board's decision-making in *inter partes* proceedings. The portions of the United States Code governing *inter partes* review proceedings use consistent and clear language to identify when a rule refers to a particular section or to the entire chapter. For example, some portions of the code refer to "this chapter," meaning Chapter 31 entitled "*Inter Partes* Review." *See, e.g.,* 35 U.S.C. §§ 311(a); 313; 314(b); 315(e); 316; 317(a); 318(a), (c). Some sections refer to other sections within that chapter, referring to those sections by number. *See, e.g.,* 35 U.S.C. § 314(a), (b). When a section refers to itself—not to the entire chapter and not to a different section—the code refers simply to "this section." *See, e.g.,* 35 U.S.C. §§ 314(d), 316(b), 317.

Section 314 read in its entirety distinguishes between "this chapter," "section 311," "section 313," and "this section." Section 314 (d) only bars review of "[t]he determination by the Director whether to institute an *inter partes* review *under this section* . . . ." 35 U.S.C. § 314(d) (emphasis added). Section 314(d) thus refers only to the Director's determination in section 314(a) of whether "the information presented . . . shows that there is a reasonable likelihood that the petitioner would prevail with respect to at least one of the claims challenged in the petition." 35 U.S.C. § 314(a). This does not bar review of decision under other sections, such as sections 311, 312, 315, 316, or 318. Thus, by its own terms section 314(a) does not apply to review of the Board's actions under the other sections of the chapter on *inter partes* review. Nor does section 314 bar judicial review of any other aspect of the agency's exercise of its authority or discretion, including its compliance with the APA.

Where, as here, the decision challenged is not about "whether to institute an *inter partes* review," i.e., whether the petitioner has established it is reasonably likely to prevail on a single ground against a single claim, but rather represents a challenge to another aspect of the Director's exercise of discretion, section 314(d) does not prohibit review.

The Board is of the view that it has discretion, not found in section 314, to deny a petition. For example, the Board has declined to institute trial on some grounds that it deems redundant in light of other instituted grounds. This sort of redundancy rejection does not reflect a factual determination that the petitioner does not have a reasonable likelihood of success on the denied grounds. *See, e.g., Volkswagen Grp. of Am. v. Farlight LLC*, IPR2013-00238, Paper 15 at 13 (P.T.A.B. Sept. 26, 2013) (instituting review for all seven challenged patent claims, but refusing to adopt "propose[d] additional grounds of unpatentability for claims 1–7 of the [challenged] patent based on Müller and Naka . . . [because] the grounds based on Müller and Naka are redundant to those based on Brown"); *Google v. B.E. Tech., LLC*, IPR2014-00031, Paper 9 at 16 (P.T.A.B. Apr. 9, 2014) (instituting review on one ground while denying the three remaining asserted grounds as "redundant to the ground of unpatentability on which we institute review for the same claims"). The Board's exercise of this discretion, not authorized by section 314, is not barred from review by section 314(d).

This reading of theplain language of section 314(d) is consistent with the Court's application of this section to institution decisions. For example, in *Cuozzo*, the Court held unreviewable the Board's allegedly improper institution of *inter partes* review on two claims where it cited prior art not cited by the petitioner on those claims. 778 F.3d at 1276. Similarly, in *In re Dominion Dealer Solutions, LLC*, the Court held unreviewable the Board's decision not to institute *inter partes* review, where the basis was the Board's finding that the petitioner did not demonstrate a reasonable likelihood of success. 749 F.3d 1379 (Fed. Cir. 2014). In both cases, the claimed error was in the substance of the institution decision.

In some cases, this Court may have applied an interpretation of section 314(d) that is broader than Hewlett-Packard's reading, suggesting that section 314(d) may bar review of decisions under other sections of Chapter 31. *See St. Jude Med., Cardiology Div., Inc. v. Volcano Corp.*, 749 F.3d 1373, 1375 (Fed. Cir. 2014) (declining to review decision not to institute *inter partes* review based on section 315(b) bar); *In re Procter & Gamble Co.*, 749 F.3d 1376, 1378 (Fed. Cir. 2014) (declining to review decision to institute *inter partes* review where patent owner argued institution was barred under section 315(a)). However, *en banc* review of this issue has been sought in *Cuozzo*. Petition for Rehearing En Banc, *In re Cuozzo Speed Techs., LLC*, No. 14-1301, Dkt. No. 37 (Fed. Cir. Mar. 23, 2015). In addition, though "a later panel is bound by the determinations of a prior panel, unless relieved of that obligation by an en banc order of the court or a decision of the Supreme Court," a later panel can come to a different conclusion where the case is factually distinguishable or where "a different governing legal issue is involved in the later appeal." *Deckers Corp. v. United States*, 752 F.3d 949, 959, 966 (Fed. Cir. 2014). None of the Court's decisions to date involve the type of rejection at issue in this appeal, and Hewlett-Packard has also raised additional issues not reached by prior panels about the Board's adherence to general principles of administrative law.

Here, review is not precluded. Section 314 does not provide for the rejection of some grounds as "redundant" over other grounds. Accordingly, the Board's redundancy rejection here is an exercise of its discretion to manage the proceedings before it. It was not a substantive decision authorized by section 314(a), because section 314(a) permits only a decision whether at least one asserted basis of invalidity of at least one claim is likely to prevail.

In addition, the Board in no event is insulated from review for abuse. "[I]t is implausible to think [Congress] intended that there be *no* forum to adjudicate statutory and constitutional challenges to regulations promulgated" by the PTO. *Bowen*, 476 U.S. at 670, 678; *see Abbott Labs. v. Gardner*, 387 U.S. 136, 140–141 (1967) ("the Court [has] held that only upon a showing of 'clear and convincing evidence' of a contrary legislative intent should the courts restrict access to judicial review") (citation omitted). This availability of review would include review for compliance with governing law. While the PTO's "interpretation of the statute under which it operates is entitled to some deference, 'this deference is constrained by [the Supreme Court's] obligation to honor the clear meaning of a statute, as revealed by its language, purpose, and history.'" *Se. Cmty. Coll. v. Davis*, 442 U.S. 397, 411 (1979) (quoting *Teamsters v. Daniel*, 439 U.S. 551, 566 n.20 (1979)). In keeping with the presumption of reviewability—especially for compliance with governing law—this Court has also recognized that review of an institution decision in an *inter partes* review proceeding may be appropriate "in situations where the PTO has clearly and indisputably exceeded its authority." *Cuozzo*, 778 F.3d at 1277.

Moreover, as set forth in Section V. below, Hewlett-Packard raises issues about the bounds of the Board's authority under the AIA to limit the scope of the final hearing, as well as similar issues under the Administrative Procedures Act. Neither is an attack on the Board's institution decision, but rather address the scope of the Board's discretion to manage the proceedings before it and the manner in which it should conduct fact finding.

C. Anticipation and Obviousness Are Distinct Grounds for Invalidity

The Court has held that anticipation is a subset of obviousness. As it explained in *Cohesive Techs, Inc. v. Waters Corp.*, "[d]espite the often quoted maxim that anticipation is the 'epitome of obviousness,' novelty under 35 U.S.C. § 102 and non-obviousness under 35 U.S.C. § 103 are separate conditions of patentability and therefore separate defenses available in an infringement action." 543 F.3d 1351, 1363 (Fed. Cir. 2008) (internal citation omitted). Anticipation and obviousness are similarly separate grounds for seeking *inter partes* review. The AIA permits review "on a ground that could be raised under section 102 or 103." 35 U.S.C. § 311(b). Here, while a finding of anticipation of claim 13 would have *de facto* provided a further conclusion of obviousness, the converse is not true.

The factual record presented by Hewlett-Packard went beyond single reference anticipation or obviousness, and included an appropriate claim of obviousness based on a combination of references. *See KSR Int'l Co. v. Teleflex, Inc.*, 550 U.S. 398, 400–401 (2007) (articulating legal standard applying to obviousness analysis).

D. The Anticipation and Obviousness Defenses at Issue Here Are Not in Fact Redundant

The obviousness ground asserted as to claim 13 was Ground 8, that claim 13 and other claims were rendered obvious by the combination of ScanJet5 with a press release about ScanJet5. *See* A2318. The Board without analysis or explanation found this ground "redundant" over the instituted grounds, namely anticipation of claims 1 to 15 by Cotte and anticipation of claims 1 to 15 by ScanJet 5. *See* A38.

For claim 13, the Petition relied upon the ScanJet5 Press Release as teaching all but one element of claim 13 (element 13.1). For element 13.1, the Petition relied upon combining the ScanJet press release with ScanJet5. *See* A2322–23; A1389–90, A1418 (¶¶ 246–351, 359). Thus, the Petition provided evidence that the elements that the Board ultimately concluded were lacking in the anticipation analysis were in fact present under the obviousness analysis. Indeed, Hewlett-Packard noted that Press Release demonstrates that even if the level of ordinary skill in the art under an obviousness analysis were determined to be quite low (e.g., the general audience for a press release of this nature), virtually all of the subject matter claimed in the ’381 patent is nonetheless disclosed in Press Release. A1424 (¶ 371). Thus, while the Board did not explain its reasoning, the Board’s conclusion that the obviousness ground, Ground 8 based on Press Release and ScanJet5, was redundant over ScanJet5 standing alone is contrary to the record evidence.

Press Release and ScanJet5 are a press release and product documentation, respectively, pertaining to the same product, the Hewlett-Packard ScanJet 5. Thus, one of skill in the relevant art would have understood that combining elements from ScanJet5 with Press Release would have been obvious to a person of ordinary skill in the art. Indeed, one of ordinary skill in the art would be inclined to read product documentation from any or all of the Hewlett-Packard products at issue here in seeking a solution to a problem. A1418 (¶ 359), A1419 (¶ 361).

Accordingly, for claim 13, the combination of these two references would have provided additional teachings not found in either standing alone, and thus the rejection of the obviousness ground as redundant of the anticipation ground on ScanJet5 was legally erroneous. Accordingly, this Court should reverse this decision and remand for further proceedings on obviousness on claim 13.

IV. THE BOARD'S AUTHORITY UNDER THE AMERICA INVENTS ACT DOES NOT INCLUDE DISCRETION TO REFUSE TO ADDRESS A PROPERLY PRESENTED GROUND FOR INVALIDITY

The plain language of the America Invents Act ("AIA") requires the Board to address all grounds that the petitioner has presented if the Board institutes trial. First, a petitioner is authorized to raise multiple grounds for invalidating the same claim. Section 322(3) of the AIA authorizes a petition that "identifies in writing and with particularity, each claim challenged, the grounds on which the challenge to each claim is based." The Trial Practice Guide confirms a petitioner's right to raise multiple grounds, stating that "a petitioner must identify each claim that is challenged and the specific statutory grounds on which each challenge to the claim is based." Office Patent Trial Practice Guide, 77 Fed. Reg. 48,756, 48,763 (Aug. 14, 2012) (codified at 37 C.F.R. pt. 42). 37 C.F.R. Section 42.104(b)(2) requires the petitioner to provide a detailed identification of the challenge, including notice of "[t]he specific statutory grounds under 35 U.S.C. [Sections] 102 or 103 on which the challenge to the claim is based and the patents or printed publications relied upon for each ground" so that the Board can adjudicate the ground.

At the time of institution, the Board must institute review if, as set out in section 314(a), there is a reasonable likelihood that the petitioner would prevail with respect to at least one of the claims challenged in the petition. This permits the Board to review the petition and act once it finds this burden of proof has been met with respect to a single claim and grounds. The Board cannot at this stage pick and choose what grounds to review; it must review the entire petition to determine if this statutory threshold has been met.

The AIA further requires that once a proceeding has been initiated, the Board is bound to issue a decision on each ground presented. It recites that "[i]f an *inter partes* review is instituted and not dismissed under this chapter, the Patent Trial and Appeal Board *shall* issue a final written decision with respect to *the patentability of any patent claim challenged by the petitioner*." 35 U.S.C. § 318(a) (emphasis added). The AIA further provides that the Director will issue a "Certificate of Patentability" of any claim "determined to be patentable." 35 U.S.C. § 307. This cannot be performed if the Board has not substantively addressed the various grounds on which the claim is asserted to be unpatentable. Absent having addressed each of the grounds, the only conclusion that the Board could reach would be that the claims were not unpatentable over the specific reference or references actually considered. Such a finding would not meet these statutory requirements. Accordingly, the AIA does not authorize the Board to conduct piecemeal review of the validity of the claims challenged by the petitioner. An agency—like the PTO—may not act "in excess of statutory jurisdiction, authority, or limitations, *or short of statutory right*." 5 U.S.C. § 706(2)(C) (emphasis added).

This reading of the plain language of 35 U.S.C. Section 318(a) is consistent with the purpose of the AIA, which is to "provid[e] quick and cost effective alternatives to litigation." H.R. Rep. No. 112-98, pt. 1, at 48 (2011). The new post grant proceedings, including *inter partes* review, are intended to "decrease[] the likelihood of expensive litigation" by "creat[ing] a less costly, in-house administrative alternative to review patent validity claims." 157 Cong. Rec. S1111 (Mar. 2, 2011) (statement of Sen. Leahy). In creating *inter partes* review proceedings, Congress's goal was "to force a party to bring all of [its] claims in one forum . . . and therefore to eliminate the need to press any claims in other fora." 154 Cong. Rec. S9989 (daily ed. Sept. 27, 2008) (statement of Sen. Kyl). In particular, it was expected that "if an *inter partes* review is instituted while litigation is pending, that review will completely substitute for at least the patents-and-printed publication portion of the civil litigation." 157 Cong. Rec. S1376 (daily ed. Mar. 8, 2011) (statement of Sen. Kyl); *accord id.* at S1361; *see also* 110 S. Rep. No. 110-259, at 71 (2008) (intending to "limit unnecessary and counterproductive litigation costs" by "tak[ing] issues off the table that cannot be resurrected in subsequent litigation"); 153 Cong. Rec. H10280 (Sept. 7, 2007) (statement of Rep. Jackson-Lee) ("the bill establishes a single opportunity for challeng[ing]" the patent and "prohibits a party from reasserting claims in court that it raised in post-grant review").

In addition, there was concern that the new *inter partes* review proceedings should not be "used as tools for harassment or a means to prevent market entry through repeated litigation and administrative attacks on the validity of a patent" because "[d]oing so would frustrate the purpose of the section as providing quick and cost effective alternatives to litigation." H.R. Rep. No. 112-98, pt. 1, at 47 (2011). In order to provide an alternative to litigation, the new *inter partes* review proceedings thus need to in the first instance reach a complete resolution of the issues raised.

Only a full resolution on the merits for an instituted trial would allow the *inter partes* review to "completely substitute for at least the patents-and-printed publication portion of the civil litigation." 157 Cong. Rec. S1376 (daily ed. Mar. 8, 2011) (statement of Sen. Kyl). Given the goal of the AIA to provide a complete resolution of issues, the Board must provide a resolution on all grounds for which there is a reasonable likelihood that the petitioner will succeed.

In addition, while the authorizing statute gives the PTO authority to manage the proceedings before it in an efficient manner, this authorization does not extend to depriving the petitioner of the review it seeks. Section 316(b) provides that "[i]n prescribing regulations under this section, the Director shall consider the effect of any such regulation on the economy, the integrity of the patent system, the efficient administration of the Office, and the ability of the Office to timely complete proceedings instituted under this chapter." While the PTO has some discretion in adopting regulations under section 316(b) to ensure efficiency, it does not have authority to refuse to apply the authorizing statute. An agency—like the PTO — may not act "in excess of statutory jurisdiction, authority, or limitations, or short of statutory right." 5 U.S.C. § 706(2)(C).

Moreover, during the legislative process, Congress made clear that the effects of estoppel should not attach unless full adjudication has occurred. The PTO was given not the ability to conduct piecemeal adjudication. Instead, the PTO's ability to limit how many petitions it considered was discussed. As a result, the Trial Practice Guide instructs that "the Board may decline to institute a proceeding where the Board determines that it could not complete the proceedings timely." Office Patent Trial Practice Guide, 77 Fed. Reg. at 48,765. This reflects "a legislative judgment that it is better that the [PTO] turn away some petitions that otherwise satisfy the threshold for instituting an inter partes or post-grant review than it is to allow the [PTO] to develop a backlog of instituted reviews" that prevents it from meeting the one-year deadline for completing proceedings. 157 Cong. Rec. S1377 (daily ed. Mar. 8, 2011) (statement of Sen. Kyl). This represents a judgment that the way to limit overall workload was to permit the PTO limit the number of petitions it reviews. Congress did not either expressly authorize or discuss authorizing the Board to undertake piecemeal review of petitions on which it did take action.

To the contrary, on the PTO's authority to set numerical limits on the new proceedings, Senator Kyl suggested that the PTO should make clear when petitions are rejected because of efficiency, in order to avoid prejudice to the petitioners: "It is understood that if the [PTO] rejects a petition during this period *because of this numerical limit*, it will make clear that the rejection was made because of this limit and *not on the merits of the validity challenges presented in the petition*." 157 Cong. Rec. S1377 (daily ed. Mar. 8, 2011) (emphasis added). Otherwise, if the PTO did not make that distinction between efficiency-based and merits-based denials,

> [E]ven a challenger with strong invalidity arguments might be deterred from using inter partes or post-grant review by fear that his petition might be rejected because of the numerical limit, and the fact of the rejection would then be employed by the patent owner in civil litigation to suggest that the experts at the [PTO] found no merit in the challenger's arguments.

Id. It would defeat this intent if the governing statutory provisions were construed as giving the Board discretion to refuse to act on portions of an otherwise meritorious challenge due to workload concerns, and then permit a challenger to argue that the final decision in fact disposed of claims not even addressed by the Board. In fact, in MPHJ's First Amended Complaint against the State of Vermont, MPHJ asserts it "wishes to send letters inquiring about infringement of its patents related to claims whose validity was recently upheld . . . following an *Inter Partes* Review." First Amended Complaint ¶ 6, *MPHJ Tech. Invs., LLC v. Sorrell*, No. 2:14-cv-00191, Dkt. No. 18 (D. Vt. Dec. 29, 2014) ("MPHJ Vt. Compl."). These "post-IPR letters" "make express reference to the results of the IPR." *Id.* In support of its alleged First Amendment Right to send these letters, MPHJ states that "[a]t least one claim of the . . . '381 Patent . . . is not invalid under Title 35 of the U.S. Code," and claims that "[o]n information and belief, at least one of the claims of the . . . '381 Patent . . . is infringed by at least one or more of the businesses in Vermont who received patent enforcement correspondence from MPHJ that is the subject of the First Amended Complaint." MPHJ Vt. Compl. ¶¶ 92, 97. In fact, on another of its patents for which a single claim recently survived *inter partes* review, MPHJ notes that it "desires now to address the widespread infringement it reasonably suspects is occurring in Vermont." *Id.* ¶ 119. This position is not warranted given the clear evidence that MPHJ's remaining claim is invalid for obviousness in light of HP's prior art scanner publications.

In an effort to reduce the burden on the Board, during the legislative process, Congress also set substantive limits on the nature of challenges that can be raised in a petition, so that they are much less complex than district court litigation. Congress originally proposed permitting a challenge on "any of the requirements for patentability set forth in sections 101, 102, 103, 112 and 251(d)." Patent Reform Act of 2005, H.R. 2795, 109th Cong. (2005) Instead, as adopted, the AIA permits *inter partes* review "only on a ground that could be raised under section 102 or 103 and only on the basis of prior art consisting of patents or printed publications." 35 U.S.C. § 311(b). Thus, the grounds the Director to consider are a limited subset of those that can be raised either in litigation or in other forms of review before the PTO.

This statutory scheme simply does not provide for a rejection on the basis that Hewlett-Packard received, which was a rejection based on Board workload rather than substance. If Congress had intended for the Board to limit its final written decision to a further subset of the petitioner's challenges, "it would have been easy enough for Congress to say so." *See Empire HealthChoice Assur., Inc. v. McVeigh*, 547 U.S. 677, 696 (2006). It did not. The Board thus simply does not have discretion to curtail proceedings in the fashion it has in this and other matters. *See, e.g., Volkswagen Grp. of Am. v. Farlight LLC*, IPR2013-00238, Paper 15 at 13 (P.T.A.B. Sept. 26, 2013) (instituting review for all seven challenged patent claims, but refusing to adopt "propose[d] additional grounds of unpatentability for claims 1–7 of the [challenged] patent based on Müller and Naka . . . [because] the grounds based on Müller and Naka are redundant to those based on Brown"); *Google v. B.E. Tech., LLC*, IPR2014-00031, Paper 9 at 16 (P.T.A.B. Apr. 9, 2014) (instituting review on one ground while denying the three remaining asserted grounds as "redundant to the ground of unpatentability on which we institute review for the same claims").

In addition, the Board recently penalized Hewlett-Packard in another proceeding, refusing to separately consider an obviousness ground not raised in a prior petition. *Dell, Inc., Hewlett-Packard Co., and NetApp, Inc. v. Elecs. & Telecomms. Research Inst.*, IPR2015-00549, Paper 10 at 5–6 (P.T.A.B. Mar. 26, 2015) (denying institution where "prior art references . . . were asserted in [a prior] IPR" and finding that "Petitioner is estopped under 35 U.S.C. § 315(e)(1) from asserting" "a ground that Petitioner could have raised in the [prior] IPR"). If this decision were held to be correct by this Court, it would not be possible to use the avenue of a second petition to cure the Board's refusal to consider all of the grounds before it in a first petition.

Accordingly, the Court should instruct the Board that, under the AIA, it should address each substantive claim of validity presented to it and fully adjudicate the validity of each patent claim challenged.

V. THE BOARD FAILED TO DISCHARGE ITS DUTIES UNDER THE ADMINISTRATIVE PROCEDURES ACT BY LEAVING UNRESOLVED ISSUES PRESENTED IN APPELLANT'S PETITION FOR *INTER PARTES* REVIEW

The Board violated the plain language of the Administrative Procedures Act by failing to substantively address in its final written decision all grounds raised by Hewlett-Packard in its petition.

A. *Inter Partes* Review Under the America Invents Act Is Subject to the Administrative Procedures Act's Standards for Formal Adjudication—Standards the Board Did Not Meet Here.

1. *Inter Partes* Review Has the Characteristics of Formal Adjudication.

Inter partes review under the America Invents Act ("AIA") should be subject to the formal adjudication requirements of the APA. The APA's formal adjudication provisions apply to an agency action that is, by statute, (1) an adjudication; (2) on the record; and (3) after a hearing. *Brand v. Miller*, 487 F.3d 862, 867 (Fed. Cir. 2007) ("Under [section] 554 of the [APA], if an agency adjudication is 'required by statute to be determined on the record after opportunity for an agency hearing,' the adjudication becomes a formal proceeding, subject to the requirements of [sections] 556 and 557 of the APA.").

Under the APA, an "'adjudication' means agency process for the formulation of an order,'" with "order" defined as "the whole or part of a final disposition, whether affirmative, negative, injunctive, or declaratory in form, of an agency in a matter other than rule making . . . ." 5 U.S.C. § 551(5), (6). Although the AIA authorized discovery and a hearing, and requires the Board to issue a final written decision, the statute does not explicitly recite that *inter partes* review proceedings are adjudicatory in nature. *See* 35 U.S.C. §§ 316, 318. As this Court has recognized, however, Congress's intent in adopting these provisions "was to 'convert[] *inter partes* reexamination from an examinational to an adjudicative proceeding,'" similar to district court proceedings. *Abbot Labs. v. Cordis Corp.*, 710 F.3d 1318, 1326 (Fed. Cir. 2013) (quoting H.R. Rep. No. 112-98, pt. 1, at 46–47 (2011) ("The [AIA] converts inter partes reexamination from an examinational to an adjudicative proceeding . . . .")). Thus, *inter partes* review proceedings are clearly adjudications under the APA.

Under the AIA, *inter partes* review must also be on the record. Though the statute does not recite the words "on the record," section 316 of the AIA "provid[es] that the file of any proceeding under this chapter shall be made available to the public . . . ." 35 U.S.C. § 316(a)(1). Since a record of the proceeding must be available to the public, it follows that the proceeding must be held *on the record*. The statute likewise "provid[es] either party with the right to an oral hearing as part of the proceeding." 35 U.S.C. § 316(a)(10). This Court's decision in *Gartside* is instructive on this point. 203 F.3d at 1313. There, the Court considered the appropriate standard of review for interference proceedings before the Board of Patent Appeals and Interferences ("BPAI")—the predecessor entity to the Patent Trial and Appeal Board. The Court held that the BPAI's *factfinding* should be reviewed under the more exacting "substantial evidence" standard—rather than the more deferential "arbitrary and capricious" standard. *Id.* at 1314. The Court held that the substantial evidence standard—which generally applies to *formal* adjudications—was appropriate in interference proceedings because, by law, the BPAI was required to conduct proceedings on the record. *Id.* That the Court has already determined that the Board's predecessor is subject to substantial evidence review and was required to hold hearings "on the record" supports the conclusion that the Board is also conducting proceedings on the record in *inter partes* review proceedings.

In *Gartside*, the Court held that the BPAI was not subject to the APA's formal adjudication requirements, however, because interference proceedings were subject to *de novo* review in a district court. 203 F.3d at 1313. In contrast, no such procedure is available here; the Board is the final fact-finder. *See* 28 U.S.C. § 1295(4)(A) (providing for Federal Circuit jurisdiction over direct appeals pursuant to 35 U.S.C. Section 319 from decisions of the Board with respect to petitions for *inter partes* review).

As the AIA requires that the Commissioner adjudicate invalidity grounds in a trial-like proceeding, requires that that the record of that proceeding be available to the public, and provides the parties a right to an oral hearing, the Court should conclude that *inter partes* review bears all the characteristics of formal adjudication under the APA.

This conclusion is supported by the legislative history of the AIA, which confirms the adjudicatory nature of *inter partes* review proceedings. The Act was intended to "convert[] *inter partes* reexamination from an examinational to an adjudicative proceeding." H.R. Rep. No. 112-98, pt. 1, at 46–47 (2011). In addition, the AIA established *inter partes* review for the purpose of "providing quick and cost effective alternatives to litigation." *Id.* at 48; *see also* 157 Cong. Rec. S1111 (Mar. 2, 2011) (statement of Sen. Leahy) (the purpose is to "decrease[] the likelihood of expensive litigation because it creates a less costly, in-house administrative alternative to review patent validity claims"); H.R. Rep. No. 112-98, pt. 1, at 47 (2011) (expressing "the purpose of the section as providing quick and cost effective alternatives to litigation").

The Board itself has underscored the adjudicatory nature of *inter partes* review proceedings, describing *inter partes* review as "neither a patent examination nor a patent reexamination," but "a trial, adjudicatory in nature [which] constitutes litigation." *Google Inc. v. Jongerius Panoramic Techs., LLC*, IPR2013-00191, Paper No. 50 at 4 (P.T.A.B. Feb. 13, 2014); *see ScentAir v. Prolitec*, IPR2013-00179, Paper 9 at 4 (P.T.A.B. Apr. 16, 2013) (describing *inter* *partes* review as "a trial, adjudicatory in nature and constituting litigation"); *Zeiss v. Nikon*, IPR2013-00362, Paper 14 at 3 (P.T.A.B. Jan. 13, 2014) ("an *inter partes* review is an adjudicatory proceeding").

2. *Inter Partes* Review Meets Tests for Formal Adjudication Under Supreme Court Law and the Law of Sister Circuits.

Direction from this Court to the Board to conduct an *inter partes* review in compliance with the formal adjudication procedures of the APA would also be consistent with case law both from the Supreme Court and sister circuits. The Supreme Court has made clear that a statute need not literally recite "on the record" in its text to impose the requirements of formal adjudication. In *Steadman v. SEC*, the Supreme Court considered whether disciplinary proceedings brought under the Investment Company Act and Investment Advisers Act were subject to the APA's formal adjudication requirements. 450 U.S. 91, 96–97 (1981). The Court noted that, while the specific statute at issue did not recite that a disciplinary proceeding needed to be conducted "on the record," "the absence of the specific phrase . . . does not make the instant proceeding not subject to [APA section] 554." *Id.* at 97 n.3. The Court held that the "on the record" requirement for the statute was "satisfied by the substantive content of the adjudication." *Id.* The Court noted that this had to be so, as the statute called for substantial-evidence review by an appellate court (which would have been frustrated in the absence of an on-the-record determination). *See id.* The Court further noted that, in the disciplinary proceedings at issue, "the Commission is required to prove violations of the securities law provisions enumerated, precisely the type of proceeding for which the APA's adjudicatory procedures were intended." *Id.*

Inter partes review has some of the same qualities as the disciplinary proceedings in *Steadman*. As in the proceedings at issue in *Steadman,* parties dissatisfied with a final written decision of the Board on *inter partes* review may appeal such decisions to this Court, suggesting that the proceedings should be on-the-record to permit review. 35 U.S.C. § 319 (providing for appeal of Board orders); 35 U.S.C. § 316(a)(1) (records of *inter partes* review proceedings must be made available to the public).

Here, the fact that the AIA was intended to make *inter partes* review adjudicatory in nature, further indicates that the "substantive content of the adjudication" is one of formal adjudication. *Steadman*, 450 U.S. at 97 n.3. The AIA explicitly requires the PTO to permit trial-like procedures, such as determining what discovery is appropriate, permitting depositions, and providing for an oral hearing. 35 U.S.C. § 316(a)(5), (6), (10). As this Court has noted, this represents a shift towards adjudicatory trial-like proceedings. *Abbott Labs.,* 710 F.3d at 1326.

Sister circuit courts have found similar proceedings to be subject to the APA's formal adjudication requirements. For example, in *Five Points Rd. Joint*

Venture v. Johanns, the court found that statutory "provision[s] for trial-type procedures," invoked formal adjudication for the National Appeals Division of the Farm Service Agency. 542 F.3d 1121, 1126 (7th Cir. 2008). There, the governing statute provided for a mandatory oral hearing once requested by a party. *Id.* at 1125–26. But the statute did not recite that the hearing must be "on the record." *Id.* The court nonetheless found APA section 554 applied to the adjudication, because, even in the absence of "those three magic words," "Congress' intent is clear." 542 F.3d at 1126. The court found this intent based on the "trial-type procedures" authorized by the governing statute and the fact that it provided for"[j]udicial review . . . upon issuance of a final determination." *Id.* Other circuits likewise have found such trial-like procedures, even in the absence of a statute requiring hearing "on the record, to indicate Congress's intent to impose formal adjudication requirements. *Friends of the Earth v. Reilly*, 966 F.2d 690, 693 (D.C. Cir. 1992) (finding "nature of the issues to be resolved in the withdrawal proceeding [to be] determinative," noting "[w]hat counts is whether the statute indicates that Congress intended to require full agency adherence to all section 554 procedural components," not whether statute contained words "on the record") (quotation and citation omitted); *Seacoast Anti-Pollution League v. Costle*, 572 F.2d 872, 876 (1st Cir. 1978), *superseded by the law of the circuit rule, as recognized in Dominion Energy Brayton Point, LLC v. Johnson*, 443 F.3d 12 (1st

Cir. 2006) ("[P]roceedings . . . conducted in order to 'adjudicate disputed facts in particular cases' . . . [are] exactly the kind of quasi-judicial proceedings for which the adjudicatory procedures of the APA were intended.") (quoting *Marathon Oil Co. v. EPA*, 564 F.2d 1254, 1262 (9th Cir. 1997)).

Consistent with the reasoning of the Supreme Court and these sister Circuits, the Court should direct the Board to comply with the APA's formal adjudication requirements in *inter partes* review.

B. The Board Violated the Administrative Procedures Act's Formal Adjudication Requirements by Failing to Issue a Final Written Decision or Hold a Hearing on All Issues Presented in the Petition.

Formal adjudication under the APA is governed by, *inter alia*, sections 556 and 557. Section 556 provides that hearings must take place before an agency or administrative law judge, and that presiding officials must be able to administer oaths, issue subpoenas, rule on evidence, take depositions, and regulate the course of a hearing. 5 U.S.C. § 556(b), (c). Under section 557, before the agency issues a decision, parties to the proceeding may submit "(1) proposed findings and conclusions; or (2) exceptions to the decisions or recommended decisions of subordinate employees or to tentative agency decisions; and (3) supporting reasons for the exceptions or proposed findings or conclusions." 5 U.S.C. § 557(c).

The record of the proceeding must, in turn, "show the ruling on *each* finding, conclusion, or exception presented" and the agency's decision "shall include a statement of . . . (A) *findings and conclusions*, and the reasons or basis therefor, on *all the material issues of fact, law, or discretion presented on the record*; and (B) the appropriate rule, order, sanction, relief, or denial thereof." *Id.* (emphasis added).

Simply put, under section 557(c), agencies "owe[] a duty to those who appear before it to address their arguments, unless frivolous." *Argo-Coller Truck Lines v. United States*, 611 F.2d 149, 152 (6th Cir. 1979). For this reason, circuit courts routinely remand agency adjudications for further proceedings where the written decision fails to address all issues presented in the underlying petition. For example, in *Argo-Coller*, the Sixth Circuit considered a challenge to an order by the Interstate Commerce Commission ("ICC") granting a petition by a trucking company, Watkins Motor Lines, to be classified as a "common carrier of foodstuffs." *Id.* at 151. The ICC granted the petition and a competitor of Watkins challenged the determination as an abuse of discretion under the APA. *Id.* The Sixth Circuit reversed the ICC, noting, among other errors, the "troubling [fact] that the ICC failed to address points of error [raised by the Petitioner] which were clearly not frivolous and which the protestants urged with vigor." *Id.* at 155. The Court remanded to the ICC for further proceedings consistent with this opinion.

Similarly, in *K & I Transfer & Storage, Inc. v. NLRB*, the Seventh Circuit considered an appeal from a National Labor Relations Board ("NLRB") order denying a request for attorneys' fees under the Equal Access to Justice Act. The General Counsel of the NLRB instituted a complaint against K & I for violations of various labor laws. After a hearing, an administrative law judge (ALJ) found no violations, and the NLRB summarily adopted the ALJ's order. *Id.* at 751. K & I appealed to the Seventh Circuit, raising, *inter alia*, the ground that the NLRB's initial order adopting the ALJ's findings and conclusions violated section 557(c) because it made "no reference at all to its request for fees and expenses for its opposition to the General Counsel's motion to amend the complaint by adding an interrogation claim against K & I's counsel." *Id.* at 753. The Seventh Circuit found that this issue "present[ed] a difficult problem," as "[t]he NLRB order "d[id] not address this claim at all" and the claim was not "a collateral contention that did not require discussion . . . ." *Id.* The court declined to hold, as the NLRB had suggested, "that the order [below] 'implicitly rejected' the[e] contention" of K & I. *Id.* The court therefore remanded "to the NLRB so that it may provide th[e] court with its 'findings and conclusions, and the reasons or basis therefore, on all material issues of fact, law, or discretion presented on the record.'" *Id.* (quoting 5 U.S.C. § 557(c)(3)(A)).

To similar effect, in *Citizens State Bank of Marshfield, Mo. v. FDIC*, the Seventh Circuit considered an appeal of an order of the Federal Deposit Insurance Corporation ("FDIC"). There, the FDIC modified an order of an administrative law judge finding violations of certain provisions of the Truth in Lending Act. 718 F.2d 1440, 1443 (7th Cir. 1983). In doing so, the FDIC failed to substantively address "exceptions" submitted by the affected parties in response to the ALJ's order. *Id.* at 1443, 1446. The Seventh Circuit reversed and remanded, finding that "5 U.S.C. § 557(c) (1976), specifically requires that the record show the FDIC's ruling on *each* exception presented by the parties." *Id.* (emphasis added). The court further found that the NLRB's merely "stat[ing] that it had considered the exceptions" and "mention[ing] that the parties submitted exceptions to the recommended decision . . . d[id] not meet the requirements of Section 557(c)." *Id.* The court directed the NLRB on remand to rule on each of the parties' filed exceptions to the ALJ's order. *Id.*

Moreover, in *Int'l Ass'n of Bridge, Structural and Ornamental Iron Workers, AFL-CIO, Local No. 111 v. NLRB*, the D.C. Circuit reviewed an NLRB order finding a violation of labor laws by a local branch of a labor union. 792 F.2d 241, 245 (D.C. Cir. 1986). The union objected on, *inter alia*, the ground that the NLRB's remedial order departed significantly from its own precedent by ordering the union to pay employees lost wages. *Id.* at 246. The court found the NLRB's order, which did not explain its basis for departing from its prior precedent, violated section 557(c) of the APA. *Id.* at 247. The court noted that the "primary purpose [of the APA's requirements for written decisions under section 557(c)] is to impose a discipline upon the agency itself, assuring that it has undergone a process of reasoned decision-making rather than haphazardly reach[ing] a result that could (on one or another basis of analysis) be sustained." *Id.* Accordingly, the court found it "unquestionably incumbent upon the [NLRB] to explain why it did not consider its decision a departure from the principles established in its prior cases, or why it considered a departure appropriate," and remanded for further proceedings in light of the opinion. *Id.* at 248.

Here, the Board declined to institute *inter partes* review of six of the eight grounds set forth in Hewlett-Packard's petition. But the Board did not substantively address the six non-instituted grounds—its *only* analysis was the following single sentence: "The additional grounds are denied as redundant in light of the determination that there is a reasonable likelihood that the challenged claims are unpatentable based on the grounds of unpatentability on which we institute an *inter partes* review." A38. Because the Board declined to institute on these grounds, it did not permit any hearing consistent with section 556 on these issues, and its final written decision—the order appealable to this Court—likewise did not address these grounds. By failing to provide a hearing on the non-instituted grounds and by failing to issue a written decision including "findings and conclusions . . . on all the material issues of fact, law, or discretion presented on the record," the Board failed to comply with the requirements of formal adjudication under the APA. 5 U.S.C. § 557. The Board's failure to address important arguments raised by Appellant in its petition—arguments that were far from "frivolous"—cannot withstand scrutiny under the APA. Accordingly, the Board's decision should be remanded with instruction for the Board to comply with the formal adjudication requirements of the APA.

C. The Board's Failure to Address Grounds in Hewlett-Packard's Appeal Also Fails Under the More Lenient Informal Adjudication Standard.

Even if the Board must only comply with more informal adjudicatory requirements of the APA, it was still required to—but did not—provide "a brief statement of the grounds for denial" of a petition submitted to it. 5 U.S.C. § 555(e). Section 555(e) requires that in even informal adjudications, "[p]rompt notice shall be given of the denial in whole or in part of a written application, petition, or other request of an interested person made in connection with any agency proceeding" and further requires that such "notice shall be accompanied by a *brief statement of the grounds for denial*." *Id.* (emphasis added). In informal adjudication the agency, in order to avoid reversal, "must examine the relevant data and articulate a satisfactory explanation for its action." *Motor Vehicle Mfrs. Ass'n of U.S., Inc. v. State Farm Mut. Auto. Ins. Co.*, 463 U.S. 29, 30 (1983). The principle underlying section 555(e) "is indispensable to sound judicial review." *Amerijet Int'l, Inc. v. Pistole*, 753 F.3d 1343, 1350 (D.C. Cir. 2014). "At bottom, an agency must explain 'why it chose to do what it did.' And to this end, conclusory statements will not do; an 'agency's statement must be one of *reasoning*.'" *Id.* (internal citation omitted, emphasis in original).

Though section 555(e) provides agencies additional latitude in comparison to the more stringent requirements of formal adjudication under section 557, the provision is not a "blank check" to deny a petition without explanation or to ignore grounds of a petition altogether, as made clear by the numerous cases reversing agencies for failure to issue written decisions in accordance with the statute. For example, in *Tourus Records Inc. v. DEA*, a petitioner challenged the Drug Enforcement Administration's ("DEA") denial of a petition to proceed *in forma pauperis* as arbitrary and capricious under the APA. The DEA denied the petition in a letter, which stated only that "the Affidavit of Indigency you submitted in lieu of a cost bond is not adequately supported." 259 F.3d 731, 737 (D.C. Cir. 2001) (internal quotation omitted). The D.C. Circuit found that this summary disposal of the petitioner's claim failed to comply with section 555(e), finding that the letter denying the petition was "not a statement of reasoning, but of conclusion." *Id.* The court concluded that "[t]he letter thus provide[d] no basis upon which we could conclude that it was the product of reasoned decisionmaking." *Id.*

Similarly, in *City of Gillette, Wyo. v. FERC*, the Tenth Circuit that found an order issued by the Federal Energy Commission ("FERC") failed to comply with the requirements for informal adjudication where the order "address[ed] none of [the petitioner's] arguments, and [the court could not] ascertain why the Commission rejected [the petitioner's] contentions." 737 F.2d 883, 886 (10th Cir. 1984). There, the petitioner appealed a denial of a permit application. *Id.* at 885. The FERC's conclusory denial of the petition, stating the "mere conclusion, tracking the terms of the regulatory criterion, that 'sufficient cause' does not exist in a particular case . . . ." was vacated and the matter remanded for further decision-making. *Id.* at 886–67.

Likewise, in *Amerijet Int'l*, the D.C. Circuit reversed an order of the Transportation Safety Administration ("TSA") for failure to comply with section 555(e). 753 F.3d at 1350. There, a shipping company sought a request for alternative screening procedures based on the fact that TSA regulations were not feasible in some circumstances. *Id.* at 1348. In a letter denying the proposals, the "TSA ignored one of Amerijet's four requests entirely, and, with respect to two of the other requests, it simply restated the rules from which Amerijet sought exceptions." *Id.* at 1350. The D.C. Circuit found that ignoring certain grounds and merely parroting legal standards in response to others failed to comply with section 555(e):

> This response, like the letter at issue in *Tourus Records*, is "not a statement of reasoning, but of conclusion." 259 F.3d at 737. It is arbitrary because it says nothing about "why" TSA made the determination. *See id.* Simply put, the May Letter had "all the

> explanatory power of the reply of Bart[le]by the Scrivener to his employer: 'I would prefer not to.'" *Butte Cnty., Cal. v. Hogen*, 613 F.3d 190, 194 (D.C. Cir. 2010) (quoting HERMAN MELVILLE, BARTLEBY, THE SCRIVNER: A STORY OF WALL STREET 10 (Dover 1990) (1853)). TSA must say more.

Id. at 1351.

Here, as a result of the Board's failure to institute on certain grounds in Hewlett-Packard's petition, the final written decision did not include a "brief statement of the grounds for denial" of each of the issues actually presented to the Board, as required under section 555(e) of the APA. Thus, the Board's actions were arbitrary and capricious, in violation of the permissive standards of informal adjudication under section 555(e). *Amerijet Int'l*, 753 F.3d at 1350.

In addition, the institution order itself does not comply with section 555(e). The one-sentence statement that certain grounds were "redundant" of others instituted is a conclusion, not a "statement of reasoning" that would permit judicial review that could confirm the finding was "the product of reasoned decisionmaking." *Id.* In sum, the Board declined to substantively address grounds in Hewlett-Packard's petition because it "would prefer not to." *Amerijet Int'l*, 753 F.3d at 1351. That is impermissible.

Accordingly, this matter should be remanded to the Board with instructions for it to comply with the informal adjudication requirement of the APA, including substantively addressing each issue presented in Hewlett-Packard's petition.

CONCLUSION AND STATEMENT OF RELIEF SOUGHT

For the reasons set forth above, the Board's decision that claim 13 was not anticipated, and the Board's refusal to undertake review of its obviousness, should both be reversed. The matter should be remanded to the Board for further proceedings. Those proceedings should be held in compliance with the standards of the Administrative Procedures Act.

Dated: May 11, 2015

Respectfully submitted,

FENWICK & WEST LLP
801 California Street
Mountain View, CA 94041
Telephone: (650) 988-8500
Facsimile: (650) 938-5200

By: */s/Charlene M. Morrow*
Charlene M. Morrow

Attorneys for Appellant
HEWLETT-PACKARD CO.

CERTIFICATE OF SERVICE

I hereby certify that on May 11, 2015, the foregoing **OPENING BRIEF OF APPELLANT HEWLETT-PACKARD CO.** to be electronically filed with the Clerk of the Court using CM/ECF, which will automatically send notification of such filing to counsel of record.

/s/Charlene M. Morrow
Charlene M. Morrow
Attorneys for Appellant

CERTIFICATE OF COMPLIANCE
WITH TYPE-VOLUME LIMITATION, TYPEFACE REQUIREMENTS, AND TYPE STYLE REQUIREMENTS

1. This brief of Appellant complies with the type-volume limitation of FED. R. APP. P. 32(a)(7)(B). The brief contains 13,105 words, excluding the parts of the brief exempted by FED. R. APP. P. 32(a)(7)(B)(iii) and FED. CIR. R. 32(b).

2. The brief complies with the typeface requirements of Federal Rule of Appellate Procedure 32(a)(5) and the type style requirements of FED. R. CIV. P. 32(a)(6). The brief has been prepared in a proportionally spaced typeface using Microsoft Office Word Version 2010 in 14-point Times New Roman.

Dated: May 11, 2015

FENWICK & WEST LLP
801 California Street
Mountain View, CA 94041
cmorrow@fenwick.com
Telephone: 650-988-8500
Facsimile: 650-938-5200

By: */s/Charlene M. Morrow*
Charlene M. Morrow
Attorneys for Appellant

ADDENDUM
TABLE OF CONTENTS

PAGE	DESCRIPTION
ADD1	Decision - Institution of *Inter Partes* Review (dated November 21, 2013)
ADD41	Final Written Decision (dated November 19, 2014)
ADD89	U.S. Patent No. 6,771,381 (dated August 3, 2004)

Trials@uspto.gov Paper 9
Tel: 571-272-7822 Entered: November 21, 2013

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

HEWLETT-PACKARD COMPANY
Petitioner

v.

MPHJ TECHNOLOGY INVESTMENTS, LLC
Patent Owner

Case IPR2013-00309
Patent 6,771,381 B1

Before SALLY C. MEDLEY, MICHAEL P. TIERNEY, and
KARL D. EASTHOM, *Administrative Patent Judges.*

EASTHOM, *Administrative Patent Judge*.

DECISION
Institution of *Inter Partes* Review
37 C.F.R. § 42.108

I. INTRODUCTION

Petitioner, Hewlett-Packard Company, filed a revised Petition requesting an *inter partes* review of claims 1-15 of U.S. Patent No. 6,771,381. Paper 6 ("Pet."). Patent Owner, MPHJ Technology Investments LLC, did not file a Preliminary Response. We have jurisdiction under 35 U.S.C. § 314.

The standard for instituting an *inter partes* review is set forth in 35 U.S.C. § 314(a):

> THRESHOLD – The Director may not authorize an inter partes review to be instituted unless the Director determines that the information presented in the petition filed under section 311 and any response filed under section 313 shows that there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition.

Pursuant to the defined threshold under 35 U.S.C. § 314(a), the Board institutes an *inter partes* review of claims 1-15 of the '381 Patent.

A. Related Proceedings

According to Petitioner, the '381 Patent is involved in a declaratory judgment action, *Engineering & Inspection Services, LLC v. IntPar, LLC*, No. 13-0801 (E.D. La., date not listed), and, with related patents, is also the subject of a consumer protection lawsuit, *Vermont v. MPHJ Tech. Investments LLC*, No. 282-5-13 (Ver. Sup. Ct. May, 2013) (MPHJ filing notice of removal to D. Vt., June 7, 2013 (No. 2:13-cv-00170)). *See* Pet. 1; Ex. 1016. The '381 Patent is related to U.S. Patent No. 7,986,426, which is the subject of another *inter partes* review, IPR2013-00302.

B. The '381 Patent

The '318 Patent describes the "Virtual Copier" (VC) system. The system enables a personal computer user to scan paper from a first device and copy an electronic version of it to another remote device, or integrate that electronic version with a separate computer application in the network. *See* Ex. 1001, Abstract.

According to the '318 Patent, "VC can be viewed as a copier. Like a copier, VC takes paper in, and produces paper going out. The only difference is that VC does not distinguish between electronic and physical paper." *Id.* at col. 71, ll. 62-65.

The VC extends from "its simplest form" to its "more sophisticated form":

> In its simplest form it extends the notion of copying from a process that involves paper going through a conventional copier device, to a process that involves paper being scanned from a device at one location and copied to a device at another location. In its more sophisticated form, VC can copy paper from a device at one location directly into a business application residing on a network or on the Internet, or [vice] versa.

Id. at col. 5, ll. 46-52.

The VC includes "five essential modules": input module, output module, process module, client module, and server module. "Each module is a counterpart to an aspect that is found on a conventional copier." *Id.* at col. 71, l. 66 – col. 72, l.1. Notwithstanding that the latter sentence refers to each module, the '318 Patent ambiguously states that "[t]here is no counterpart to VC's Server Module on a conventional copier." *Id.* at col. 72, ll. 59-60. In any event, the other four modules have "counterparts" on "conventional" copiers: "The Input Module manages paper or electronic paper entering VC. . . . The counterpart to VC's Input Module on a conventional copier is the scanner subsystem." *Id.* at col. 72, ll. 5-13. "The Output Module manages paper or electronic paper exiting VC. . . . The counterpart to

VC's Output Module on a conventional copier is the printer or fax subsystem." *Id.* at ll. 14-23. "The Process Module applies processing to the electronic paper as it is being copied. . . . The counterpart to VC's Process Module on a conventional copier is the controller." *Id.* at ll. 24-34. "The Client Module presents the electronic paper as it is being copied, and any relevant information related to the input or output functions. . . . The counterpart to VC's Client Module on a conventional copier is the panel." *Id.* at ll. 34-45. "Unlike conventional copiers, VC's Server Module is a unique subsystem that can communicate with the other modules as well as third-party applications." *Id.* at ll. 44-47.

Figure 28 of the '381 patent, reproduced below, represents an embodiment of VC:

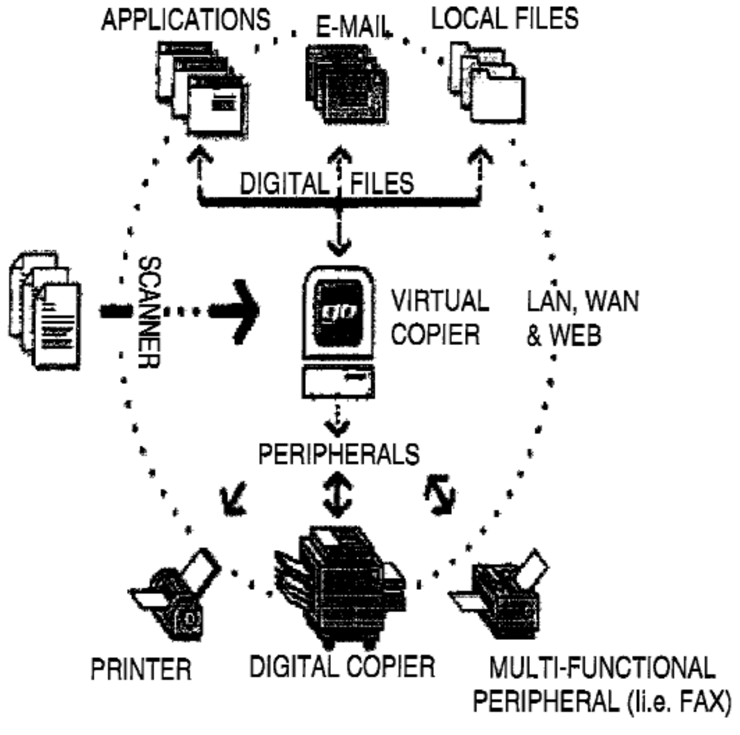


FIG. 28

Figure 28 depicts various peripheral devices attached to a Virtual Copier on a network. *See id.* at Abstract.

C. Exemplary Claim

Of the challenged claims, claims 1 and 12-15 are independent. Challenged claim 1 follows:

> 1. A computer data management system including at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet, comprising:
>
> at least one memory storing a plurality of interface protocols for interfacing and communicating;
>
> at least one processor responsively connectable to said at least one memory, and implementing the plurality of interface protocols as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications, wherein said software application comprises at least one of:
>
> at least one input module managing data comprising at least one of paper and electronic paper input to the computer data management system, and managing at least one imaging device to input the data through at least one of a scanner and a digital copier, and managing the electronic paper from at least one third-party software applications;
>
> and at least one module communicable with said at least one input, output, client, and process modules and external applications, and capable of dynamically combining the external applications with at least one of digital capturing devices and digital imaging devices.

D. References Relied Upon

Petitioner relies upon the following references:[1]

Dow, U.S. Patent No. 6,611,291 B1 (Aug. 26, 2003, filed Aug. 7, 1998) (Ex. 1010);

Cotte, U.S. Patent No. 5,499,108 (Mar. 12, 1996) (Ex. 1011);

HP Network ScanJet 5 Scanner User's Guide (2nd ed. October, 1997) (Ex. 1006, "SJ5");

Technical Support Solutions Guide, HP ScanJet 4Si Scanner (1995) (Ex. 1007, "SJ4Si");

HP 9100C Digital Sender User Guide (2nd ed. 2001) (Ex. 1008, "9100C");

HP LaserJet 3100 Product User's Guide (1st ed. April 1998) (Ex. 1009 "HP3100"); and

"HP Introduces Next-Generation Network Scanner" (1997) (Ex. 1015, "SJ5PR").

E. The Asserted Grounds

Petitioner asserts the following grounds of unpatentability under 35 U.S.C. §§ 102 and 103:

Claims 1-15 as anticipated under 35 U.S.C. § 102(b) by SJ5;

Claims 1-12 as anticipated under 35 U.S.C. § 102(a) by SJ4SI;

Claims 1-15 as anticipated under 35 U.S.C. § 102(a) by HP3100;

[1] The '381 Patent claims priority by continuation to U.S. Provisional Application 60/108,798 (filed November 13, 1998), and by continuation-in-part to several provisional applications (filed October 18, 1996). Ex. 1001, col. 1, ll. 7-31. However, Petitioner maintains that the Office did not recognize priority to October 18, 1996. Pet. 3. The Office records, along with the face of the '381 Patent, indicate that November 13, 1998 is the effective priority date. The Board proceeds pursuant to that indication.

Claims 1-15 as anticipated under 35 U.S.C. § 102(a) by HP9100C;

Claims 1-15 as anticipated under 35 U.S.C. § 102(e) by Dow;

Claims 1-15 as anticipated under 35 U.S.C. § 102(b) by Cotte;

Claims 1-4, 6, 8, 10, and 14 as anticipated under 35 U.S.C. § 102(b) by SJ5PR;

Claims 5, 7, 9, 11-13, and 15 as obvious under 35 U.S.C. § 103(a) over SJ5PR and SJ5. Pet. ii.

II. ANALYSIS

A. Claim Construction

In an *inter partes* review, "[a] claim in an unexpired patent shall be given its broadest reasonable construction in light of the specification of the patent in which it appears." 37 C.F.R. § 42.100(b); *see also Office Patent Trial Practice Guide*, 77 Fed. Reg. 48756, 48766 (Aug. 14, 2012) (Claim Construction). Under the broadest reasonable construction standard, claim terms are given their ordinary and customary meaning as would be understood by one of ordinary skill in the art in the context of the entire disclosure. *In re Translogic Tech., Inc.*, 504 F.3d 1249, 1257 (Fed. Cir. 2007). Any special definition for a claim term must be set forth in the specification with reasonable clarity, deliberateness, and precision. *In re Paulsen*, 30 F.3d 1475, 1480 (Fed. Cir. 1994). In the absence of such a special definition or other consideration, "limitations are not to be read into the claims from the specification." *In re Van Geuns*, 988 F.2d 1181, 1184 (Fed. Cir. 1993).

The Board construes the following claim phrases and terms:

At least one, at least one of, and related phrases

Claim 1 and most of the other claims recite the phrase "at least one" or "at least one of" in a number of places. For example, claim 1 recites "*at least one* input module managing data comprising *at least one of* paper and electronic paper input to the computer data management system, and managing *at least one* imaging device to input the data through *at least one* of a scanner and a digital copier, and managing the electronic paper from *at least one* third-party software applications" (emphases added).

The phrase "at least one input module" means "one or more input modules." *See Rhine v. Casio, Inc.*, 183 F.3d 1342, 1345 (Fed. Cir. 1999) ("Use of the phrase 'at least one' means that there could be only one or more than one."). Petitioner proposes that a related type of phrase, "at least one of A and B," means "at least one of A or B." *See* Pet. 4. Under some situations, according to *Superguide Corp. v. DirecTV Enters. Inc.*, 358 F.3d 870, 886 (Fed. Cir. 2004), the plain meaning of "at least one of A and B" is "at least one of A and at least one of B." Quoting a "common treatise on grammar," *Superguide* focuses on an example wherein the preposition "in" precedes a list (i.e., "'[i]n spring, summer, or winter' means 'in spring, in summer, or in winter'"), and reasons that the phrase "'at least one of,' modifies each member of the list, i.e., each category in the list." *Id.* (quoting example in William Strunk, Jr. & E.B. White, *The Elements of Style* 27 (4th ed. 2000) (brackets from *Superguide*)). However, *Superguide* points out that the specification involved there does not enlarge the scope of the plain meaning, and reasons that each term in the list embraces a different category, each of which must take on a chosen value: "Every disclosed embodiment teaches that the user must choose a value for each designated category." *Id.* at 887 ("Importantly, the flow chart uses a conjunctive criteria list, i.e., the system's user must choose at least one value for each designated criteria, or the logic would be inoperable.").

Accordingly, *Superguide* has been distinguished on the basis that the normal conjunctive meaning does not apply when the specification or claims imply a broader meaning. *See Joao v. Sleepy Hollow Bank*, 348 F. Supp. 2d 120, 124 (S.D.N.Y. 2004) (a conjunctive reading of the phrase, "wherein the banking transaction is at least one of a clearing transaction, a check clearing transaction, an account charging transaction, and a charge-back transaction," would be nonsensical because a single banking transaction cannot be all four).[2]

Following the principles outlined *supra*, the claim 1 phrase, "at least one input module managing data comprising at least one of paper and electronic paper," is reasonably broad enough to be read in the alternative. For example, the phrase encompasses one input module managing data from electronic paper, such as from a software application. The claims and Specification do not invoke a conjunctive reading for that phrase or similar phrases. For example, claim 1 does not reference, necessarily, different categories of paper, or different categories of electronic paper.

Moreover, the Specification of the '381 Patent indicates the intent to treat different inputs and outputs, and perform the other recited functions, in the alternative, using separate input modules for each type of input, and separate output modules for each type of output. In other words, at least one or more modules perform at least one or more of certain functions, and each module is tailored specifically to one type of device or application:

[2] At least one practitioner describes an established contrary view of the plain meaning prior to *Superguide*, which published after the earliest possible effective filing date of the '381 Patent: "It is therefore better practice to avoid the word 'or.' Several accepted techniques for doing this were developed in the past. One was to recite 'at least one of element A and element B,' which is equivalent to 'or' but avoids the troublesome word itself." Allen Wood, *Drafting Patent Claims for use in the United States in Mechanical and Electrical Cases*, at 23 (2003), http://www.awoodpatents.com/claims_booklet_(rev._nov_28__03).pdf.

> [I]n order to support outputting to a third-party application, an Output Module is developed that is unique to that third-party application. Likewise, an Input Module is developed that is unique to a third-party application in order to support reading images from that application.
>
> It is the optional Input and Output Modules that render VC extendable. For each third-party application there is a unique pair of Input and Output Modules that understand the third-party application, and how to copy images to and from that application. . . . In this way[,] Virtual Copier can grow indefinitely, to support any number of third-party applications.
>
> The significant point is that the Input and Output Modules have their own interface, and can be developed independently from any other module. As long as the input and output Module conform to the API specified in this document it will plug-and-play with VC. VC will be able to mix and match the custom Input and Output Module with its standard and other custom Input and Output Modules.

Ex. 1001, col. 9, ll. 19-39.

Other examples refer to modules and their functions in the alternative: "The Input Module manages paper *or* electronic paper entering VC. This module manages imaging devices to input paper through, scanners, MFPs, *or* the new breed of digital copiers. The Input Module also manages reading electronic paper from third-party *or* proprietary applications." *Id.* at col. 8, ll. 6-11 (emphases added). The Specification also states that the Virtual Copier's "GO button can copy paper, whether physical or electronic, from one device and[/]or application to another device and/or application." *Id.* at col. 6, ll. 44-46. In other words, the Specification consistently reveals an intent to treat choices alternatively, and in some cases, blurs distinctions, by grouping "and" and "or" together.

Accordingly, the claim 1 phrase, "at least one input module managing data comprising at least one of paper and electronic paper input to the computer data management system, and managing at least one imaging device to input the data through at least one of a scanner and a digital copier, and managing the electronic paper from at least one third-party software application," is interpreted to embrace one or more input modules each managing one or more of the recited functions. The managing functions may overlap.

Claim 1 also recites the phrase "at least one module communicable with said at least one input, output, client, and process modules and external applications, and capable of dynamically combining the external applications with at least one of digital capturing devices and digital imaging devices." The phrase reasonably requires the "at least one module" to be communicable with at least one input module, output module, client module, or process module, or external applications. As noted, the term "at least one" means "one or more," and the preposition "of" is not recited in this claim phrase, unlike the claims at issue in *Superguide.* Hence, the above-listed phrase in claim 1 means one or more modules communicable with one or more input, output, client, or process modules, or external applications.

The Specification supports the interpretation, by stating that the server module functions to create a variety of systems in the alternative, as follows:

> Server Module-Unlike conventional copiers, VC's Server Module is a unique subsystem that can communicate with the other modules as well as third-party applications. . . . A virtual copier can be created with VC by combining a scanner with a printer, or by combining a scanner with an application; or by combining an application with an image printer. . . . There is no counterpart to VC's Server Module on a conventional copier.

Ex. 1001, col. 8, ll. 44-60.

In general, phrases of the type "at least one of A and B," appear throughout the claims and Specification, usually in terms of functions performed by "one or more modules." Based on the foregoing discussion, unless otherwise noted, at this juncture, phrases of the type discussed here, "at least one of A and B," and "at least A and B," are interpreted in the alternative, i.e., "one or more A or B."

Third-party software application

The terms "third-party software application," or "applications," recited in claim 1, and other claims, do not preclude software that resides in printers, scanners, or other devices. The Specification refers to "third-party" software as "proprietary" software. *See* Ex. 1001, col. 8, l. 11. It also refers to "business applications (such as Microsoft Office, Microsoft Exchange, Lotus Notes)." *See id*. at col. 5, ll. 56-57; col. 46, ll. 19-21. The Specification also refers to copying paper "from one device and[/]or application to another device and/or application," thereby broadly blurring any distinction between a device and a device having a software application. *See id*. at col. 6, ll. 44-46. Therefore, the terms mean a program that may or may not be on a device.

Managing

Claim 1 requires that the input module manages data. The Specification does not specify what "managing," in the context of data, means. Managing may include "conventional copier . . . scanner subsystem" commands. *See id*. at col. 8, ll. 13-14. In other words, managing may require receiving or transferring the data, and possibly, but not always, transforming the data to conform to a specific format. As noted in the discussion of the phrase "at least one of," some disclosed modules are tailored as a specific plug-and-play modules, indicating that each module may perform a custom transform function. *See also* Ex. 1001, col. 9, ll. 22-24 (input module "is unique to a third-party application in order to support reading images from that application"). Therefore "managing" means sending or employing signals to facilitate receiving or transmitting data, or transforming data, or both.

Module

Claim 1 recites a "computer data management system" comprising "at least one input module," "at least one module," and "at least one input, output, client, and process modules and external applications." Petitioner does not propose a definition for "module."

One plain meaning of "module" is "[a] distinct and identifiable unit of a computer program for such purposes as compiling, loading, and linkage editing." MCGRAW-HILL DICTIONARY OF SCIENTIFIC AND TECHNICAL TERMS 1285 (5th Ed. 1994) (Ex. 3001). Another plain meaning of "module," which is similar, but broader slightly, is "a logically separable part of a program. *Note:* The terms 'module,' 'component,' and 'unit' are often used interchangeably or defined to be sub-elements of one another in different ways depending upon the context. The relationship of these terms is not yet standardized." IEEE 100 THE AUTHORITATIVE DICTIONARY OF IEEE STANDARDS TERMS SEVENTH EDITION 704 (2000), *available at* http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4116801 (last visited Sept. 19, 2013).

As noted *supra*, the '381 Patent states that input and output modules are unique to each third-party printer or scanner application, and "understand the third-party application, and how to copy images to and from that application." Ex. 1001, col. 49, ll. 59-61. The '381 Patent also states that "[t]he Client Module is generally simply an interface to the Server Module." *Id*. at col. 50, ll. 15-16. As also noted *supra*, the modules also have "counterparts" in prior art copier or scanner systems. In other words, modules may include other modules and may overlap in functionality.

In addition, the '381 Patent states that the modules all "support COM-based interfaces for simple and direct support from all major Windows development environments." Ex. 1001, col. 9, ll. 59-61. On the other hand, the '381 Patent indicates that the "standard COM component" constitutes a mere example, and that modules can have different structure:

> The computer architecture is implemented, *for example*, as a <u>*standard COM component*</u>, as an ActiveX control; the specifications designed by Microsoft, published in the technical literature, and incorporated herein by reference. ActiveX control (COM) support is currently available within any Microsoft 32-bit Windows operating environment. ActiveX controls are supported by all OLE-based applications, including all of Microsoft's end-user products (e.g., Microsoft Office, Word, Access, Powerpoint, Access), the main Internet Browsers (Microsoft's Internet Explorer and Netscape's Navigator--the latter with an add-in product and by 4Q97 directly), most other name-brand end-user Windows products (e.g., Lotus Notes), and all major development environments (e.g., Microsoft Visual Basic and Visual C++, Delphi, Borland C++, Power Builder). By implementing the architecture as, for example, an ActiveX control, complex technologies can be programmed by virtually any Windows or Intranet user or developer. *Of course, other component specifications may also be used*.

Id. at col. 53, ll. 30-48 (emphasis added). In addition to embracing "standard COM components," and "other component specifications," the '381 Patent embraces "many other languages (e.g. Java) and distributed architectures (e.g., COBRA)." *Id*. at col. 53, ll. 49-53. The '381 Patent also indicates that typically, in the prior art, "[e]very engine, such as text retrieval or an OCR (Optical Character Recognition) engine, has a unique interface. This interface is generally a 'C'-level API (Application Program Interface)." *Id*. at col. 53, ll. 54-57. However, the '381 Patent does not specify that each module must have a unique or a generic interface. Claim 10 supports this interpretation, by specifically claiming "at least one server module application programmer interface (API)."

According to the foregoing discussion, the '381 Patent is consistent with both ordinary meanings of a module. Therefore, each "module," as recited in the claims, is a logically separable part of the claimed data management system, and a module may include another module and overlap with another module in functionality.

Seamlessly

Claim 3 recites the phrase "wherein the computer data management system includes the capability to integrate an image using software so that the image gets seamlessly replicated and transmitted to one of other devices and applications." The '381 Patent Specification refers to a "simple solution," delivering "paper processing to existing Intranet and client-server business processes without any fuss," so that "an office clerk" can "easily copy a report from a desktop scanner to the company's Intranet-networked copier." Ex. 1001, col. 46, ll. 42-47. In light of the Specification, the term "seamlessly" means "a low amount of effort," or "easily."

Go operation

Claim 5 recites "wherein the computer data management system includes an interface that enables copying . . . using a single 'GO' operation." The '381 Patent describes a "GO operation" as similar to a "START" (button) operation on a conventional copy machine. Ex. 1001, col. 46, l. 66 – col. 47, l. 3. Further, "[t]his GO button can copy paper, whether physical or electronic, from one device and[/]or application to another device and/or application," *id*. at col. 47, ll. 7-9, and "the user simply has one sequence to execute: select From, select To, and then press GO," *id*. at ll. 30-33. In light of the Specification, the term means "an operation that begins a process."

Modules object, program object, document object, and system management event object

Claim 11 recites "COM-based interfaces" including "at least one modules object maintaining a first list of available input, output, and process modules." The '381 Patent does not provide a definition for a modules object. The '381 Patent states that "a preferred embodiment . . . has, for example, the following structure illustrated in FIG. 36[;] however, *alternative structures and/or functionality may optionally be used for this object and/or other objects used in the present invention*." Ex. 1001, col. 75, ll. 8-12 (emphasis added). Figure 36 portrays a box with the following text in the box: "Collections of Copier[]Module objects, of types Input[]Module, Output[]Module, and Process[]Module respectively." No apparent structure is depicted.

Accordingly, a "modules object" has "alternative structures and/or functionality" and represents a program or file. The phrase "at least one modules object maintaining a first list" means one or more programs, files, or other structures, each of which can store, or point to, a list or portions of a first list. (The term "maintain" is discussed below.)

Claim 11 also recites "at least one program object maintaining a second list," "at least one document object maintaining information," and "at least one system management event object used to provide feedback." Figure 36 also depicts five boxes, without specifying any type of structure, with the boxes labeled as follows: "Object," "Collection," "Property," "Method," "Event." These objects, according to the '381 Patent, as noted *supra*, also have "alternative structures and/or functionality." Therefore, all the claimed objects include similar definitions: one or more programs, files, or structures, for performing the designated functions.

Means-plus-function limitations, maintain

Claims 9 and 13 each recite a "server module" that includes the same four means-plus-function limitations. Claim 9 depends from claim 7, which depends from claim 1, and recites "wherein the server module includes" the four means-plus-function limitations. However, "the server module" recited in claim 9 lacks antecedent basis. For purposes of this proceeding, "the server module" recited in claim 9 is interpreted either to refer back to "at least one module communicable," which is recited in claim 1, or to refer to an additional module, a server module.

In general, the '381 Patent describes the server module as follows: "a scheduler of activities, providing the information and initiating the modules at the appropriate time in the virtual copy operation. The Server Module manages the other Modules. It does not know about the internal workings of the modules, nor the contents of the information being copied." Ex. 1001, col. 74, ll. 44-49.

The server module recited in dependent claim 9 and independent claim 13, includes, *inter alia*, the following four means-plus-function limitations:

> enable virtual copy operation means for initiating, canceling, and resetting said computer data management system;
>
> maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used . . . , said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules;
>
> maintain currently active modules means for maintaining said input, output, and process modules currently being used, . . . and saving the currently active modules in a process template file; and
>
> maintain complete document information means for maintaining information . . . and saving the information in a document template file.

As claimed, the word "maintain" precedes three of the means clauses. As disclosed, and as discussed *supra*, a modules object in the server module maintains a list of input, process, and output modules. Ex. 1001, col. 74, ll. 54-55. The word "maintain," in the context of the claims, and normally, means "[t]o preserve or retain" for a certain time period. *See* THE AMERICAN HERITAGE DICTIONARY OF THE ENGLISH LANGUAGE 787 (1975). Therefore, claims 9 and 11 generally recite three preserving, keeping, or retaining means clauses.

These means-plus-function limitations invoke 35 U.S.C. § 112, ¶ 6.[3] Petitioner asserts that "virtually no structure is provided corresponding to these means." Pet. 6. Apparently, Petitioner refers to a lack of algorithmic structure.

Regarding "enable virtual copy operation means for initiating, canceling, and resetting said computer data management system," recited in claim 9, the '381 Patent generally describes that "[t]he Server Module supports simple methods that accomplish the basic copier functionality of go, cancel, and reset." Ex. 1001, col. 78, ll. 13-15. The '381 Patent also indicates that the reset function involves returning to default settings. *Id.* at col. 82, ll. 38-42. The discussion *supra* of the Go operation similarly shows that the corresponding structure encompasses basic copier structure, including known algorithms associated with the known hardware structure. Accordingly, the corresponding structure for the enable copy operation means clause includes the "basic" buttons, processor, and memory to process and store known begin, stop, and reconfigure algorithms; i.e., corresponding to known structure in prior art copiers or scanners.

[3] Section 4(c) of the Leahy-Smith America Invents Act ("AIA") re-designates 35 U.S.C. § 112, ¶ 6, as 35 U.S.C. § 112(f). Because the '381 Patent's filing date antecedes September 16, 2012, the effective date of AIA, the pre-AIA version of 35 U.S.C. § 112 applies.

In general, the common corresponding structure for each of the final three means clauses recited in claims 9 and 13 appears to include memory for storing specific programs, lists, or information, in the nature of files, or registries. *See* Ex. 1001, col. 74, ll. 8-33. As indicated *supra*, the '381 Patent discusses objects in the context of structure that appears to be, on this record, associated somewhat with the functions involved here, and the '381 Patent embraces broad structure for the objects (*see* construction of objects *supra*). Therefore, the "maintain list of available module means" corresponds to a generic listing algorithm and memory that stores, or points to, a list of modules. The "maintain currently active module means" corresponds to a generic storing algorithm and memory that stores, or points to, the active input, output, and process modules. The "maintain complete document information means" corresponds to a generic algorithm and memory that stores, or points to, information regarding a current document or file being copied.

Patent Owner will have an opportunity, in its Patent Owner Response, to inform the Board as to its construction of the means-plus-function (and other) limitations or to forgo that opportunity, leaving the Board with the intrinsic record and Petitioner's construction. Any claim construction of a means-plus-function should set forth the corresponding structure disclosed in the specification that performs the claimed function, including any computer or microprocessor, computer program, and algorithm. *WMS Gaming, Inc. v. Int'l Game Tech.*, 184 F.3d 1339, 1349 (Fed. Cir. 1999) ("In a means-plus-function claim in which the disclosed structure is a computer, or microprocessor, programmed to carry out an algorithm, the disclosed structure is not the general purpose computer, but rather the special purpose computer programmed to perform the disclosed algorithm.").

Merely referencing a specialized computer, or some undefined component of a computer system, or elements that are essentially black boxes designed to perform the recited function, will not be sufficient, because there must be some explanation of how the computer or the computer component performs the claimed function. *See Blackboard, Inc. v. Desire2Learn, Inc.*, 574 F.3d 1371, 1383-85 (Fed. Cir. 2009); *Net MoneyIN, Inc. v. VeriSign, Inc.*, 545 F.3d 1359, 1366-67 (Fed. Cir. 2008).

Claims 8 and 12, certain phrases

Claim 8 recites "wherein the one or more of the external devices and applications integrates the computer data management system into an external application via one of running the computer data management system, as an external service and embedding the computer data management system as an embedded service." Based on the claim language, "one of" refers to running and embedding, and means one of running or embedding. The claim does not define a relative internal system for the "external" application. Claim 8 depends from claim 1 and requires that the computer data management system includes a memory and a processor. It is not clear how that hardware portion of the claimed management system can be integrated by embedding or running anything. Accordingly, based on this record, integrating by one of embedding and running, means that the computer data management system exists as, runs as, or was built as, part of another (i.e., external) application.

Claim 12 recites "[a] computer data management system" that includes a "single function copy operation linking devices," "a one step programming method, . . . a method of recreating a module oriented copier in software," and "a copier interface." Claim 12 recites phrases that appear to place the claim into two statutory categories, "process" and "machine," which is impermissible under 35 U.S.C. § 101. For purposes of this proceeding, the "one step programming method" is interpreted to describe functional characteristics of a machine, and includes a method of making the "electronic business processes" in the machine, so that the system (a machine) includes "a module oriented copier in software," and was made "with no or minimal reprogramming." Typically, however, as is the case here, how the system was made is not afforded patentable weight, because nothing in the Specification indicates that after it has been made, it would behave differently than a system that required more than minimal reprogramming.

B. Asserted Grounds of Unpatentability

1. Cotte – Anticipation, Claims 1-15

Relying on the Wibbels Declaration, Ex. 1005, Petitioner reads the elements of claims 1-15 onto Cotte's scanning network system. Pet. 42-50. Cotte generally describes an integrated system that produces scanned document data that can be sent to a host computer and e-mailed, faxed, or printed. Ex. 1011, Abstract, Fig. 10, col. 10, ll. 42-58. Cotte's paper input device

> senses the insertion of a document to be scanned, initiates a host computer process, i.e., controls the host process by insertion of the paper and symbols on the paper, scans the images and text on the paper, . . . send[s] the scanned data to the host for further electronic processing such as display, transmission, storage or modification. Principally, this new technology is a paper input device using scanning technology which controls the host computer rather than the other way around.

Id. at col. 2, ll. 42-51.

Cotte also discloses that

> the user can put the document in the paper input device and the input device software will automatically scan the document, send the data to the host in any of the ways described herein, and the input device software resident on the host will then cause a pop-up window to appear on the screen where the image of the scanned document appears.

Id. at col. 16, ll. 58-63.

One way to send data involves symbol recognition software: "the input device includes symbol recognition software that can recognize symbols on the document to be scanned which indicate where the document is to be FAX'ed, sent as E-mail, etc." *Id*. at col. 11, ll. 27-31.

According to the '381 Patent, "VC can be viewed as a copier," even though "VC does not distinguish between electronic and physical paper." Ex. 1001, col. 71, ll. 62-65. It follows, based on the foregoing description of the two systems, that the two systems are similar, at least in terms of the functionality as a basic copier operating on electronic and physical paper.

The preamble of claim 1 follows:

> A computer data management system including at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to . . . one or more of external devices and applications . . . connectable at least one of locally and via the Internet.

Cotte discloses the preamble, according to the discussion *supra* and Petitioner's showing. For example, as described *supra*, Cotte's system manages the host computer and sends an image and text to an external e-mail system, or to a fax system, which shows that Cotte's system is "connectable at least one of locally and via the Internet," as the claim 1 preamble recites. *See* Pet. 43; Ex. 1011, Fig. 10.

Claim 1 also recites the following limitations: "at least one processor responsively connectable to said at least one memory, and implementing the plurality of interface protocols as a software application" and "at least one memory storing a plurality of interface protocols for interfacing and communicating." Cotte discloses microprocessor 352 implementing software for controlling operations of the input device that is stored in ROM/RAM 132. *See* Ex. 1011, Fig. 11A. For example, the microprocessor controls scan inputs and outputs between fax inputs and outputs. Accordingly, as Petitioner and Mr. Wibbels explain, Cotte's processor, which implements this stored software for fax and input communications, satisfies the listed limitations. *See* Pet. 43; Ex. 1005 ¶ 298; Ex. 1011, col. 8, ll. 39-55; col. 10, ll. 32-55.

Claim 1 also recites

> at least one input module managing data comprising at least one of paper and electronic paper input . . . , and managing at least one imaging device to input the data through at least one of a scanner and a digital copier, and managing the electronic paper from at least one third-party software applications.

To satisfy the input module limitation, Petitioner relies on Cotte's scanner, input device 114 at Figure 11A, which includes the stored fax protocol discussed *supra*. Pet. 43; Ex. 1005 ¶ 298. Specifically, in Cotte, the "input device software controlling the[ ]microprocessor 352 includes software routines to send the appropriate commands to the Fax modem to control its operations in sending the scanned data as a Fax." Ex. 1011, col. 9, ll. 8-12. Therefore, because managing data includes controlling the transfer of the data pursuant to a conventional scanner subsystem, as discussed in the Claim Construction section, Cotte's input device software routines manage paper and electronic paper, through the scanner, an imaging device, thereby satisfying at least the first two recited functions of the input module.

Cotte also implies that input software routines automatically control data between the host and the fax modem, in both directions. *See* Ex. 1011, col. 8, ll. 39-56. Controlling this fax data transfer also satisfies the first, second, and third input module functions, the third being "managing the electronic paper from at least one third-party software application[].” Further, Cotte also discloses that the “input device includes symbol recognition software that can recognize symbols on the document to be scanned which indicate whether the document is to be FAX’ed, sent as E-mail, etc.” Ex. 1011, col. 11, ll. 27-31. The claimed input module functions also read on that process, because the symbol recognition software helps to manage electronic paper from a scanner using symbol software, which constitute a third-party application.

Claim 1 also recites “at least one module communicable with said at least one input, output, client, and process modules and external applications, and capable of dynamically combining the external applications with at least one of digital capturing devices and digital imaging devices.” Petitioner maintains that Cotte discloses the following: “A module in communication with an input module. Upon receiving a scanned image, other modules attach a scanned image to a clipboard, an email, or send it as a fax.” Pet. 43 (citing Ex. 1011, col. 10, ll. 43-53, FIG. 17, and Ex. 1005 ¶ 295).

At the cited passage, Cotte discloses host software, which includes

> a drop down menu 250 presenting options to the user regarding what should be done with the scanned image. These menu options can be such things as “FAX this image” as symbolized by icon 253 or “Send this image as an E-mail message” as symbolized by icon 255, or “Send this image to the laser printer for printing” as symbolized by icon 257 . . . .

Ex. 1011, col. 10, ll. 43-50.

Cotte’s drop down menu teaches “at least one module communicable with said at least one input . . . modules,” such as the input module identified *supra*. The menu (including its program) is “capable of dynamically combining” the external e-mail or fax application with at least digital capturing devices or digital imaging devices, such as, for example, the scanner.

In a related manner of recognizing symbols on scanned paper to further implement faxing, printing, storing, or e-mailing thereof, Cotte's system provides for the input device software "to invoke specific types of software commonly found on user's computers or to invoke specific 'macros' or predefined sequences of instructions." Ex. 1011, col. 11, ll. 47-49. These macros also represent, under one alternative, "at least one module communicable with said at least one input" module. The input device software also corresponds to at least one communicable module, even if it contains or invokes other modules, as discussed *supra* in the Claim Construction section.

The symbols, software for creating same, the e-mail system, or fax system, comprise external applications, which are capable of being combined with "at least one of digital capturing devices and digital imaging devices," as claim 1 also requires.

Pursuant to the foregoing discussion, Petitioner establishes a reasonable likelihood of prevailing on the ground of unpatentability of claim 1 as anticipated by Cotte.

Claims 2-7 depend from claim 1. Claim 2 further requires "a printer, a facsimile, and a scanner." Cotte discloses these devices as discussed *supra,* and as Petitioner discusses. *See* Pet. 44.

Claim 3 requires "the capability to integrate an image using software so that the image gets seamlessly replicated and transmitted to at least one of other devices and applications, and via the Internet." Mr. Wibbels notes that Cotte discloses sending a scanned image "automatically" in an e-mail; therefore, Cotte provides seamless replication and transmission, and skilled artisans would have understood that this e-mail capability also implies Internet capability. *See* Ex. 1005 ¶¶ 291, 297; Pet. 44. Petitioner establishes a reasonable likelihood that claim 3 reads on Cotte's system.

Claim 4 further requires the capability to integrate images into a destination application without modifying that application. Mr. Wibbels relies on destination applications that satisfy the limitation, including e-mail, fax, clipboard storage, printing, and insertion into a drawing file. *See* Ex. 1005 ¶¶ 295-296; Pet. 44; Ex. 1011, col. 11, ll. 23-53; col. 21, ll. 6-15.

Claim 5 further requires enabling copying images using "a single 'GO' operation." Petitioner sufficiently shows that claim 5 reads on Cotte's drop down menu icons, described *supra*, or the other buttons, described next. *See* Pet. 44.

Claim 6 recites a similar limitation, "at least one of electronic document and paper processing with a single programming step." Petitioner essentially relies upon Cotte's copy button for that limitation. *See* Pet. 44 (citing Ex. 1011, col. 19, ll. 40-42; col. 20, ll. 43-52, 55-60, Fig. 17; and Ex. 1005 ¶ 300). Figure 17, cited by Petitioner, displays a number of single operation buttons, including "Photocopy" button 257. Figure 22 similarly discloses a copy button 310. Pressing the button causes the system to scan the image and send it to a printer for copying. *See* Ex. 1011, col. 19, ll. 42-48. Accordingly, Petitioner sufficiently shows that claim 6, like claim 5, reads on Cotte's drop down menu icons or the other buttons.

Claim 7 recites "wherein the software application comprises: at least one output module . . . at least one process module . . . ; and at least one client module." Each module performs certain functions, which pertain to the module type. As indicated *supra* in the Introduction, the '381 Patent states that the modules have "counterparts" in conventional copier systems, thereby indicating that functions performed by the modules may be conventional.

Addressing the output module functions, Petitioner cites to Cotte's system, which "[m]anage[s] an input scanner to receive data that is faxed to another computer or system," and "[s]oftware output . . . used to communicate data to various external devices, e.g., sending scanned data to a printer." Pet. 45. Claim 7 recites the following one or more output modules:

> at least one output module managing the data output from the computer data management system, managing at least one imaging device to output the data to at least one of a standard Windows printer, an image printer, and a digital copier, and managing the output of the data to the third-party software application.

Petitioner's citations appear to rely on Cotte's software that manages data output from a scanner and outputs it to a printer or faxes it to another computer or system, for example, as E-mail, or third-party software. Pet. 45 (citing Ex. 1011, col. 8, ll. 38-52; col. 10, ll. 49-50; col. 18, ll. 52-55). Petitioner also points to block 328 in Figure 23, which has the following label: "send image data to laser printer." The preceding block, block 326, has the following label: "scan image of document." Block 328 represents at least one output module that manages data output from the system, by managing data from at least one imaging device, the scanner, to output data to a laser image printer. Block 328 also represents an output module that manages the output of the data to the third-party software application, because a laser printer implicitly includes third-party software, as discussed *supra* in the Claim Construction section. Similar to the discussion of the input module, the output module may perform some data transformation to conform to a specific device and/or application, or may perform functions akin to the "conventional copier . . . printer or fax subsystem." Ex. 1001, col. 8, ll. 22-23.

In addition, or alternatively, Cotte discloses block 344 at Figure 24, which includes a "configuration preferences file," which, in turn, dictates what each menu button does, by checking a "script." Choices include "'send this as a FAX,'" or "'insert this scanned image into drawing file XXXX REV 2.'" Ex. 1011, col. 21, ll. 10-12. Such block, file, or script further comprises "at least one output module" for "managing the output of the data to the third-party software application," such as a facsimile application or drawing file application.

Claim 7 also recites "at least one process module." Petitioner maintains that gray scale pixel processing, or scanning according to instructions on a scanned paper, read on the recited module(s). *See* Pet. 45. Cotte discloses "the job of the recognition software portion of the input device software" as recognizing symbols and translating the symbols into commands for later processing by another "proper software package." Ex. 1011, col. 13, ll. 8-22. The recognition or translation portions of the recognition software, or the proper software package, corresponds to at least one process module that performs the "at least one data processing" and "additional functionality" as recited in claim 7. Cotte's gray scale pixel processing, and data compression, respectively represented in boxes 264 and 266 at Figure 18, also correspond to a process module or modules, which "apply[ ] multiple processes to a single virtual copy," as claim 7 further requires. *See* Ex. 1011, col. 17, ll. 5-30.

Claim 7 also recites "at least one client module" presenting the paper or electronic paper data and "information related to at least one of the input and output functions." Petitioner relies on Cotte's pop-up displays, which display the incoming electronic paper data, as reading on these client modules. Pet. 45. Figure 17 displays a typical pop-up menu showing different options for incoming pages. Ex. 1011, col. 16, ll. 60-66; col. 18, ll. 50-55. Claim 7 requires the client module to present the electronic paper data, and information related to at least one of the input and output functions. Displaying the pages constitutes presenting paper and electronic paper data, because the data reveals relative gray scale data values or document types, etc., while portraying written output options for archiving, photocopying, mailing, and faxing, constitutes providing the requisite input and output function information. *See* Ex. 1011, col. 17, ll. 5-10, Fig. 17. Petitioner sufficiently shows that Cotte anticipates claim 7.

Claim 8 essentially requires the computer management system to be integrated into an external application with an external device and application by running the system, as discussed *supra*, in the Claim Construction section. Petitioner points to an embodiment that includes part of the input device software running on the host computer. *See* Pet. 45-46 (citing Ex. 1011, col. 10, l. 60 – col. 11, 1. 5; col. 18, ll. 46-51; Ex. 1005 ¶ 305). Another embodiment has the computer management software stored inside of RAM and running with a processor in the input device. Ex. 1011, col. 8, ll. 37-39. Therefore, the computer management system software is integrated with an external application and external device and application by running part of it on the host and part of it on the external scanning device. *See* Ex. 1011, Fig. 11A. Petitioner sufficiently shows that Cotte anticipates claim 8.

Claim 9 depends from claim 7 and recites a server module that includes four means-plus-function limitations, as discussed *supra.* Claim 13 is similar to claim 9 and also recites a server module that includes the same four means-plus-function limitations. Cotte's input device software includes copy button and other typical algorithms. *See* Ex. 1011, Fig. 23 (box 320 states "has photocopy button been pushed?"). The input device software, or the drop-down menus discussed *supra*, correspond to "server module" recited in claims 9 and 13. Modules, as noted *supra*, including the server module, may include other modules.

Addressing the first recited means-plus-function limitation in claim 9, "enable virtual copy operation means for initiating, canceling, and resetting said computer data management system," Petitioner points to the means and functions disclosed in connection with Figures 13A and 23 of Cotte. *See* Pet. 46. Block 330, part of the input software, represents initiating, timeout, and default determinations, which respectively correspond to the three recited functions. *See* Ex. 1011, col. 19, ll. 56-61; col. 20, ll. 23-48; Fig. 23.

The enable copy means reads on Cotte's algorithms and hardware, because the '381 Patent embraces known structure, and includes a copy button and associated software, according to the Claim Construction and Introduction *supra*.

The second recited means-plus-function limitation follows: "maintain a list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used, . . . said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process, and server modules." Petitioner points to a menu of options as a list on a hard disk "(used at startup of a copier) and RAM (used during operation of a copier) to 'determine what software packages are resident.'" Pet. 46. Mr. Wibbels also declares that a skilled artisan would have understood in a Windows implementation that the "menu of options" would be stored in files that would be maintained in a registry. Ex. 1005 ¶ 308. Petitioner's showing is reasonable. Cotte's system checks the hard disk and the RAM to generate menu options. *See* Ex. 1011, col. 15, ll. 36-42; *see* Ex. 1005 ¶¶ 308, 320.

The third recited means-plus-function requires the currently active modules to be maintained in a program object, and saving said modules in a process template file. Petitioner and Mr. Wibbels essentially indicate that this means-plus-function clause reads on a normal Windows implementation, which includes template files. *See* Pet. 46; Ex. 1005 ¶¶ 308-309. This appears reasonable on this record, because the currently active modules necessarily would be the object of the software implementing it, including the input software, and the modules would be stored on the computer. As indicated *supra*, corresponding structure for such broad functions appears to be an accessible portion of memory and known algorithms for accessing the memory. Petitioner makes a similar showing for the final recited means-plus-function clause. *See* Pet. 46; Ex. 1005 ¶¶ 308-310. Accordingly, Petitioner's demonstrates a reasonable likelihood that Cotte anticipates claims 9 and 13.

Claim 10 depends from claim 7 and requires a server module to "include[] at least one server module application programmer interface (API)." According to Mr. Wibbels, Cotte specifically discloses Macintosh applications as an example and also contemplates other applications, including "a Windows operating system using corresponding Windows components, such as COM-based interfaces." Ex. 1005 ¶ 312 (citing Ex. 1011, col. 1. l. 24; col. 2, l. 8; col. 23, ll. 19-22; col. 8, ll. 13-15; Fig. 10; col. 10, ll. 43-53; Fig. 17). The cited passages, including the generic reference to "[m]any computers," support the declared theory that skilled artisans would have contemplated that Cotte discloses "a Windows operating system," because such systems were well-known, and according to the '381 Patent, include "standard COM components." Ex. 1001, col. 53, l. 31; *see also* the module discussion in the Claim Construction section. Accordingly, Petitioner demonstrates a reasonable likelihood that Cotte anticipates claim 10.

Claim 11 depends from claim 10 and recites "at least one server module" that includes lists of available and currently selected modules, and document and program objects, which all appear to be implicit, based on this record, in a prior art system, such as Cotte's. *See* Pet. 47; Ex. 1005 ¶¶ 294-317. The recited lists are similar to the lists discussed *supra* in connection with claim 9. Petitioner demonstrates a reasonable likelihood that Cotte anticipates claim 11.

Independent claim 12, recites, *inter alia*, a system claim that includes "single function copy operation linking devices; . . . a one step programming method . . . with no or minimal reprogramming . . . , a method of recreating a module oriented copier in software; [and] . . . a copier interface." The limitations are similar to those discussed *supra* in connection with claims 1-5 and 10. Accordingly, Petitioner relies on its showings with respect to claims 1-5 and 10. *See* Pet. 47-48.

For example, Cotte's drop down menu, which provides options and single step copies and transmissions, and other features as discussed in connection with claims 1 and 5, read on the claimed "copier interface" and "single function copy operation linking devices." *See* Pet. 47-48; Ex. 1005 ¶¶ 299, 319, Fig. 17.

The recited "method" steps are construed, as noted *supra*, as defining the final structure and including process limitations as to how the system has been built – resulting in a modular system that was built with minimal reprogramming. Cotte's system employs modules and minimally impacts the host system, as explained *supra*, and according to Mr. Wibbels. *See* Ex. 1005 ¶ 295. As another example, Cotte discloses that "the manufacturer can 'pretrain' the symbol recognition software . . . to be shipped with the software for the input device to invoke specific types of software commonly found on user's computers or to invoke specific 'macros' or predefined sequences of instructions." Ex. 1011, col. 11, ll. 44-49. In other words, the shipped software can be implemented on a host as a single step, with minimal or no reprogramming to the host, to invoke software commonly found on the hosts. In addition, after Cotte's system has been built one time, it requires no *reprogramming*. Finally, according to the Claim

Construction section, how the system was built does not create a structural distinction.

Claims 14 and 15 recite elements and phrases that are the same as or substantially similar to the elements and phrases discussed above. Petitioner points to its showings with respect to claims 1, 5, 10, and 12 and also relies on the Wibbels Declaration. *See* Pet. 49-50 (citing Ex. 1005).

Accordingly, Petitioner establishes a reasonable likelihood of prevailing on the ground that Cotte anticipates claims 1-15.

2. SJ5 – Anticipation, Claims 1-15

SJ5 describes "[t]he HP Network ScanJet 5 scanner[, which] can scan items such as memos, letters, brochures, photographs, newspaper clippings, and advertisements and store and distribute them electronically." Ex. 1006, 11. It works on a network with one or more personal computers, including Microsoft Windows based computers. *See* Ex. 1006, 12.

SJ5 further discloses the following system capabilities:

> Scanned documents are sent to the destinations you select at the scanner control panel using public and private destination lists. When you send a scanned document, it can arrive at one or more of the following destinations:
>
> - The application you have designated as your *inbox* in the HP Network ScanJet 5 Utility. From the inbox, it can be further distributed to other users via e-mail, and so forth, imported as a graphic into other applications or read into word processing applications using the OCR feature.
> - The inbox of another registered user or multiple registered users.
> - A fax machine.
> - An Internet e-mail address.
>
> If you have the PaperPort software installed on your computer or on

> your network, you can send a scanned document directly to the application you have designated in your automatic workflow in the HP Network ScanJet 5 Utility. In addition, at the scanner control panel you can select the printer to which to copy a scanned document.

Ex. 1006, 18.

In other words, SJ5 describes a system that is similar to the disclosed invention. Claim 1 recites "at least one memory storing a plurality of interface protocols for interfacing and communicating" and "at least one processor responsively connectable to said at least one memory, and implementing the plurality of interface protocols as a software application for interfacing and communicating with the plurality of external destinations including one or more of the external devices and applications." To show the external destinations, Petitioner points to OCR software disclosed in SJ5. Pet. 9 (citing Ex. 1006, 12, 93, 97; Ex. 1005 ¶ 117). As to the interface protocols, Petitioner relies on "[i]nstall software," instructions for installing software, such as Utility, PaperPort, and OCR software. According to SJ5, the install program is somewhere on the network or on an installation CD. Ex. 1006, 12. SJ5 describes the installation process as follows: "You can install either from the network or from the installation CD." *Id.*

Petitioner appears to rely on inherency, essentially asserting that the SJ5 system's installation software must be stored in memory with interface protocols. *See* Ex. 1005 ¶ 117; Pet. 8. Mr. Wibbels declares that "to apply the process of OCR, the ScanJet 5 requires a processor in communication with the memory that is storing the software." Ex. 1005, ¶ 117. The system includes a personal Microsoft Windows 386 or 486 computer. Ex. 1006, 12. At this juncture, it appears that to use software in a network, including fax, e-mail, OCR, and other applications, as SJ5 discloses, protocols necessarily must be stored and processed by a processor. Therefore, SJ5 discloses at least one processor that invokes installation software and also "implement[s] the plurality of interface protocols . . . for interfacing and communicating with the plurality of external destinations," as claim 1 requires. *See* Ex. 1005 ¶ 119 (citing Ex. 1006, 47).

Claim 1 also recites "at least one module communicable with said at least one input, output, client, and process modules." According to Mr. Wibbels, "one skilled in the art would understand that a system that links to external applications would include a module capable of combining the digital imaging device with the linked external applications." Ex. 1005, ¶ 119 (citing Ex. 1006, 47). Software corresponding to the linking function, and the corresponding Link Bar, constitutes at least one communicable module. *See* Pet. 12 (discussing the PaperPort Link Bar as corresponding to the server module recited in claim 10).

Software implicit in the scanner corresponds to at least one input module. Software implicit in a printer, fax, e-mail or other applications corresponds to at least one output module. *See* Pet. 9. The scanner control panel, or the work station inbox, correspond to the at least one client module. *See* Ex. 1006, 15, 18. Icons associated with Link Bar software, including icons and workflow processes associated therewith, correspond to the at least one process module. For example, the OCR module processes scanned documents, and another implicit software module converts scanned spreadsheets into "editable numbers." *See id.* at 89. Adobe^{TM} Acrobat^{TM} Reader constitutes another process module. *See id.* at 12. Therefore, Petitioner sufficiently shows that SJ5 anticipates claim 1.

Claim 2 further recites "a printer, a facsimile, and a scanner." SJ5 discloses these devices as discussed *supra,* and as Petitioner discusses. Pet. 9; Ex. 1006, 18. Petitioner sufficiently shows that SJ5 anticipates claim 2.

Claim 3 requires "the capability to integrate an image using software so that the image gets seamlessly replicated and transmitted to at least one of other devices and applications, and via the Internet." The ScanJet 5 Utility software allows scanned devices to be sent to an inbox and "distributed to other users via e-mail" or "imported as a graphic into other applications or read into word processing applications using the OCR feature." Ex. 1006, 18. Users can "scan documents to an Internet e-mail address." *Id*. at 19. Petitioner also relies on "automatic workflow[s]" implemented in the PaperPort software to show that the process is seamless. *See* Pet. 9; Ex. 1006, 35. On this record, Petitioner shows that claim 3 reads on the SJ5 system.

Claim 4 further requires the capability to integrate images into a destination application without modifying that application. The discussion *supra* in connection with claim 3 applies to claim 4. Petitioner also refers to using icons on the Link Bar. Pet. 9; Ex. 1006, 89. On this record, Petitioner shows that claim 4 reads on the SJ5 system.

Claim 7 recites at least one output module, at least one process module, and at least one client module, including certain functions associated therewith. These modules are discussed *supra* in connection with claim 1. SJ5 discloses the certain functions associated with these modules, where, as noted, SJ5 discloses a similar system to that described in the '381 Patent, and the '381 Patent discloses that the modules have counterparts in prior art copiers and scanners. *See* Pet. 10. Petitioner sufficiently shows that SJ5 anticipates claim 7.

Claims 9 and 13 each require a "server module" that includes means-plus-function limitations that involve maintaining lists or other functions, or objects involving lists, as discussed *supra*. Claim 11 is similar and recites a "server module" that includes objects maintaining lists and information and performing copy functions. Petitioner refers to a PaperPort Link Bar as corresponding to at least one server module API (application programmer interface). *See* Pet. 12.

According to SJ5, all program files are stored "on the network and will be shared with others in your workgroup." Ex. 1006, 14. Sharing implies an accessible list of available programs to share. Also, the system contemplates accessible copies: "Registered users can create private destination lists by copying destinations from the public destination list and creating their own destinations." *Id*. at 16. A display on the scanner control panel in SJ5 displays user and destination lists. *Id*. at 15. The destination lists include a wide variety of inputs, outputs and workflows as described *supra*. Also, the system contemplates accessible copies: "registered users can create private destination lists by copying destinations from the public destination list and creating their own destinations." *Id*. at 16.

With further respect to the lists in claims 9, 11, and 13, Petitioner refers to "an output destinations list in ScanJet 5 Utility." Pet. 12. Mr. Wibbels also explains that ScanJet 5 utility provides for the creation of automatic workflows. Ex. 1005 ¶ 137. According to Mr. Wibbels, these features imply to skilled artisans that SJ5 provides "a list of available input, output, and process modules to be stored in the system." *Id.* Mr. Wibbels further describes that lists of active processes and programs would be stored in a registry as part of disclosed troubleshooting methods. *See id*. at ¶ 132.

Based on the foregoing discussion, Petitioner sufficiently shows that PaperPort software and Scan Jet software constitute at least one server module that includes the claim elements recited in claims 9, 11, and 13, which for the most part, at this juncture, appear to embrace well-known copy management structure, according to the broad disclosures in the '381 Patent and the discussion *supra*. *See id*. at ¶¶ 130-148. Petitioner and Mr. Wibbels describe how SJ5 reads on the remaining elements of claims 9, 11, and 13.

Claim 14 recites limitations that are similar to the limitations recited in claim 1. Relying on the Wibbels Declaration, Petitioner persuasively discusses the remaining claims, including claims 5, 6, 8, 10, and 12. *See* Pet. 10-13. Based on the foregoing discussion, and considering that SJ5 discloses Windows-based modules and a system that is similar to the disclosed invention, Petitioner establishes a reasonable likelihood of prevailing on the ground that SJ5 anticipates claims 1-15. *See* Pet. 8-15 (citing Ex. 1005).

3. Remaining Asserted Grounds of Unpatentability

Petitioner asserts additional grounds of unpatentability, as listed in Section I.E., *supra*. The additional grounds are denied as redundant in light of the determination that there is a reasonable likelihood that the challenged claims are unpatentable based on the grounds of unpatentability on which we institute an *inter partes* review. *See* 37 C.F.R. § 42.108(a).

III. CONCLUSION

The Petition demonstrates a reasonable likelihood of prevailing on the following grounds of unpatentability: anticipation of claims 1-15 by Cotte and by SJ5.

IV. ORDER

In consideration of the foregoing, it is hereby

ORDERED that pursuant to 35 U.S.C. § 314, an *inter partes* review is hereby instituted as to claims 1-15 of the ’381 Patent for the following grounds of unpatentability:

1. Claims 1-15 for anticipation by Cotte; and
2. Claims 1-15 for anticipation by SJ5;

FURTHER ORDERED that no other grounds of unpatentability set forth in the Petition are authorized for the *inter partes* review as to claims 1-15 of the ’381 Patent;

FURTHER ORDERED that pursuant to 35 U.S.C. § 314(d) and 37 C.F.R. § 42.4, notice is hereby given of the institution of a trial that will commence on the entry date of this decision; and

FURTHER ORDERED that an initial conference call with the Board is scheduled for 10:00 AM ET on Dec. 17, 2013. The parties are directed to the *Office Trial Practice Guide*, 77 Fed. Reg. 48756, 48765-66 (Aug. 14, 2012) for guidance in preparing for the initial conference call, and should be prepared to discuss any proposed changes to the Scheduling Order entered herewith and any motions the parties anticipate filing during the trial.

For Petitioner:

Stuart Meyer
Jennifer Bush
Fenwick & West LLP
Smeyer@fenwick.com
jbush@fenwick.com

For Patent Owner:

Scott Horstemeyer
N. Andrew Crain
THOMAS | HORSTEMEYER, LLP
scott.horstemeyer@thomashorstemeyer.com
andrew.crain@thomashorstemeyer.com

Trials@uspto.gov
Tel: 571-272-7822

Paper 35
Entered: November 19, 2014

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

HEWLETT-PACKARD CO.,
Petitioner,

v.

MPHJ TECHNOLOGY INVESTMENTS, LLC,
Patent Owner.

Case IPR2013-00309
Patent 6,771,381 B1

Before MICHAEL P. TIERNEY, KARL D. EASTHOM, and GREGG I. ANDERSON, *Administrative Patent Judges.*

EASTHOM, *Administrative Patent Judge*.

FINAL WRITTEN DECISION
35 U.S.C. § 318(a) and 37 C.F.R. § 42.73

I. INTRODUCTION

Petitioner, Hewlett-Packard Co., filed a (resubmitted) Petition requesting *inter partes* review of claims 1–15 of U.S. Patent No. 6,771,381 (Ex. 1001). Paper 6 ("Pet."). Patent Owner, MPHJ Technology Investments, LLC, did not file a (non-required) Preliminary Response, and we instituted *inter partes* review of claims 1–15, on two grounds of unpatentability, as listed below. *See* Paper 9 ("Dec. on Inst.").

Subsequent to institution, Patent Owner filed a Patent Owner Response (Paper 20, "PO Resp."), and Petitioner filed a Reply (Paper 25, "Pet. Reply") thereto. Substantively, Petitioner relies on a declaration by Mark Wibbels (Ex. 1005), and Patent Owner relies on a declaration by Glenn Weadock (Ex. 2002). Patent Owner deposed Mr. Wibbels. Ex. 2003. The parties requested and appeared at an oral hearing before the panel, which transpired on August 18, 2014. The record includes a transcript of the hearing. Paper 34 ("Tr.").

We have jurisdiction under 35 U.S.C. § 6(c). This Final Written Decision, issued pursuant to 35 U.S.C. § 318(a) and 37 C.F.R. § 42.73, addresses issues and arguments raised during trial.

For the reasons that follow, we determine that Petitioner has met its burden of proving, by a preponderance of the evidence, that claims 1–12, 14, and 15 of the '381 Patent are unpatentable. Petitioner, however, has not demonstrated by a preponderance of the evidence that claim 13 of the '381 Patent is unpatentable.

A. Related Proceedings

According to Petitioner, the '381 Patent is involved in a declaratory judgment action, *Engineering & Inspection Services, LLC v. IntPar, LLC*, No. 13-0801 (E.D. La., Oct. 10, 2013), and, with related patents, is also the subject of a consumer protection lawsuit, *Vermont v. MPHJ Tech. Investments LLC*, No. 282-5-

13 (Ver. Sup. Ct. May, 2013) (MPHJ filing notice of removal to D. Vt., June 7, 2013 (No. 2:13-cv-00170)). *See* Pet. 1; Ex. 1016. The '381 Patent application is a grand-parent to U.S. Patent No. 7,986,426, which is also the subject of an *inter partes* review. *See Ricoh Americas Corp. v. MPHH Tech. Invs., LLC*, Case IPR2013-00302 (PTAB) ("'302 IPR").

B. The '381 Patent

The '381 Patent describes a "Virtual Copier" (VC) system. The system enables a personal computer user to scan paper from a first device and copy an electronic version of it to another remote device, or integrate that electronic version with a separate computer application in the network. *See* Ex. 1001, Abstract.

According to the '381 Patent, "VC can be viewed as a copier. Like a copier, VC takes paper in, and produces paper going out. The only difference is that VC does not distinguish between electronic and physical paper." *Id.* at col. 71, ll. 62–65.

The VC extends from "its simplest form" to its "more sophisticated form":

> In its simplest form it extends the notion of copying from a process that involves paper going through a conventional copier device, to a process that involves paper being scanned from a device at one location and copied to a device at another location. In its more sophisticated form, VC can copy paper from a device at one location directly into a business application residing on a network or on the Internet, or [vice] versa.

Id. at col. 5, ll. 47–54.

The VC includes "five essential modules": input module, output module, process module, client module, and server module. "Each module is a counterpart to an aspect that is found on a conventional copier." *Id.* at col. 71, l. 66 – col. 72, l. 1. Notwithstanding that the latter sentence refers to each module, the '381 Patent ambiguously states that "[t]here is no counterpart to VC's Server Module on a conventional copier.” *Id*. at col. 72, ll. 59–60. In any event, the other four modules have “counterparts” on “conventional” copiers: “The Input Module manages paper or electronic paper entering VC. . . . The counterpart to VC’s Input Module on a conventional copier is the scanner subsystem.” *Id*. at col. 72, ll. 5–13. “The Output Module manages paper or electronic paper exiting VC. . . . The counterpart to VC’s Output Module on a conventional copier is the printer or fax subsystem.” *Id*. at ll. 14–23. “The Process Module applies processing to the electronic paper as it is being copied. . . . The counterpart to VC’s Process Module on a conventional copier is the controller.” *Id*. at ll. 24–34. “The Client Module presents the electronic paper as it is being copied, and any relevant information related to the input or output functions. . . . The counterpart to VC’s Client Module on a conventional copier is the panel.” *Id*. at ll. 34–45. “Unlike conventional copiers, VC’s Server Module is a unique subsystem that can communicate with the other modules as well as third-party applications.” *Id*. at ll. 44–47.

Figure 28 of the ’381 Patent follows:

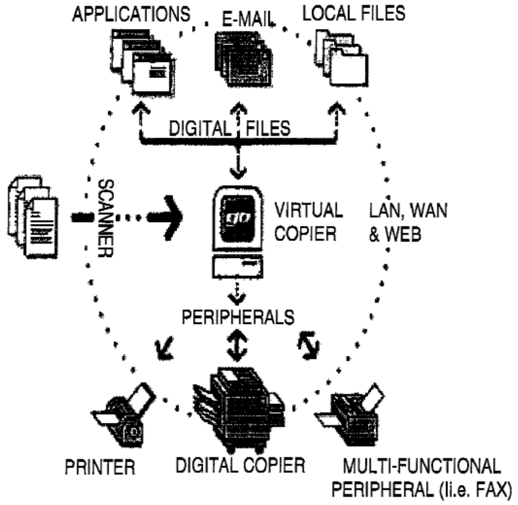


FIG. 28

Figure 28 depicts various peripheral devices attached to a VC on a network. *See id*. at Abstract.

C. Illustrative Claim

Of the challenged claims, claims 1 and 12–15 are independent. Challenged claim 1 follows:

> 1. [1.P] A computer data management system including at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet, comprising:
>
> [1.1] at least one memory storing a plurality of interface protocols for interfacing and communicating;
>
> [1.2] at least one processor responsively connectable to said at least one memory, and implementing the plurality of interface protocols as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications, wherein said software application comprises at least one of:
>
> > [1.3] at least one input module managing data comprising at least one of paper and electronic paper input to the computer data management system, and managing at least one imaging device to input the data through at least one of a scanner and a digital copier, and managing the electronic paper from at least one third-party software applications; and
> >
> > [1.4] at least one module communicable with said at least one input, output, client, and process modules and external applications, and capable of dynamically

> combining the external applications with at least one of digital capturing devices and digital imaging devices.

See Pet. 9 (bracketing by Petitioner), 44 (same).

D. The Grounds

We instituted trial on the following grounds:

Claims 1–15 as anticipated under 35 U.S.C. § 102(b) by SJ5.[1]

Claims 1–15 as anticipated under 35 U.S.C. § 102(b) by Cotte.[2]

II. ANALYSIS

A. Claim Construction

In an *inter partes* review, "[a] claim in an unexpired patent shall be given its broadest reasonable construction in light of the specification of the patent in which it appears." 37 C.F.R. § 42.100(b); *see also Office Patent Trial Practice Guide*, 77 Fed. Reg. 48,756, 48,766 (Aug. 14, 2012) (Claim Construction). Under the broadest reasonable construction standard, claim terms are given their ordinary and customary meaning as would be understood by one of ordinary skill in the art in the context of the entire disclosure. *In re Translogic Tech., Inc.*, 504 F.3d 1249, 1257 (Fed. Cir. 2007). Any special definition for a claim term must be set forth in the specification with reasonable clarity, deliberateness, and precision. *In re Paulsen*, 30 F.3d 1475, 1480 (Fed. Cir. 1994). In the absence of such a special definition or other consideration, "limitations are not to be read into the claims from the specification." *In re Van Geuns*, 988 F.2d 1181, 1184 (Fed. Cir. 1993).

[1] HEWLETT PACKARD, HP NETWORK SCANJET 5 SCANNER USER'S GUIDE (2d ed. 1997) (Ex. 1006).

[2] U.S. Patent No. 5,499,108 (Mar. 12, 1996) (Ex. 1011).

At least one, at least one of

Claim 1 and most of the other claims recite the phrase "at least one" or "at least one of" in a number of places. For example, claim 1 recites "[1.2] . . . wherein said software application comprises at least one of: [1.3] at least one input module managing data . . .; and [1.4] at least one module communicable with said at least one input, output, client, and process modules and external applications."

In our Institution Decision, we interpreted phrases of the type "at least one of A and B" and "at least A and B" in the alternative, i.e., "one or more A or B." Dec. on Inst. 11. The parties do not challenge this interpretation. Patent Owner agreed with it during the oral hearing, stating that "as Petitioner stated, these are alternative claim elements. Patent Owner takes no issue with that interpretation, that either - - that claim element 1.3 [at least one input module] and 1.4 [at least one module communicable] are claimed in the alternative." Tr. 31:6–9 (*see supra* claim 1). Patent Owner's declarant, Mr. Weadock, also agrees with this general construction outlined in the Petition. *See* Ex. 2002 ¶ 17.

Software Application/Application

As indicated above, claim 1 recites "said software application comprises at least one of: at least one input module . . . and at least one module communicable with" other modules. Claim 1 also recites "one or more of external devices and applications," and "external applications." Patent Owner contends that an "application" is "a discrete software program executable on an operating system for the purpose of accomplishing a task." PO Resp. 6. Patent Owner also contends that an "application" and a "software application" do not include "firmware": "While firmware is made up of software, it is not the same thing as a software application. Nowhere in the specification of the '381 patent is 'application' or 'software application' used in the context of device firmware." *Id*

Petitioner contends that Patent Owner's interpretation is too narrow. *See* Pet. Reply 2–3. Petitioner points out that the '381 Patent describes "'VC is in one embodiment . . . *optionally* a standalone application.'" Pet. Reply 3 (quoting Ex. 1001, col 8, ll. 66–67, emphasis by Petitioner). The Specification supports a broader interpretation than Patent Owner urges in other places. For example, it shows specific examples of an application or software application that are not limited to a discrete software program and do not preclude firmware: "an application (e.g., Lotus Notes, Microsoft Exchange, *the Internet*, or *an electronic filing system*)." Ex. 1001, col. 6, ll. 59–61 (emphasis added). Also, VC can copy "in and out of devices and business applications (such as Microsoft Office, Microsoft Exchange, Lotus Notes)." *Id*. at col. 46, ll. 19–20.

Patent Owner essentially contends that an "application" and a "software application" mean the same thing. *See* PO Resp. 6–7. This interpretation renders the term "software" redundant. The term "application" is not limited to software, as the disclosed examples of the Internet and electronic filing systems verify. We found in the Institution Decision that the '381 Patent Specification "refers to copying paper 'one device and[/]or application to another device and/or application,' thereby broadly blurring any distinction between a device and a device having a software application." Dec. on Inst. 12 (quoting Ex. 1001, col. 6, ll. 44–46). Claim 8, which depends from claim 1, provides for "integrat[ing]" or "embedding the computer data management system," which includes the "software application," into an "external application." This claim 8 phrase further shows that an application includes hardware. Therefore, based on the disclosure, including specific examples, an "application" may include hardware, software, or software and hardware.

As to a software application, notwithstanding Patent Owner's arguments, Patent Owner does not distinguish "firmware" from stored software that is part of a distributed software application. The Specification includes distributed architecture with the VC software stored virtually anywhere. For example, the Specification states that "[t]he VC software can reside on a PC, LAN/WAN server, digital device (such as a digital copier), or on a web server to be accessed over the Internet." Ex. 1001, col. 46, ll. 21–24.

Patent Owner also contends that a "software application" means a "single software application." PO Resp. 18–19. To support the argument, Patent Owner reasons that a "software application" precludes "firmware": "It would not make sense to say that the firmware of a scanner and host software make up the same (discrete) software application." *Id.* at 19. Contrary to the arguments, the claims do not recite the word "single" or "discrete," or otherwise preclude firmware.

The '381 Patent Specification also does not support a "software application" as limited to a "single" or "discrete" software application. Examples of a broader meaning abound in the '381 Patent Specification: "Accounting systems, like most business applications, typically have no way of maintaining an electronic copy of the physical invoice and adding a document management system to an accounting system is cumbersome . . . and . . . difficult to coordinate." Ex. 1001, col. 47, ll. 51–56. This disclosure equates a "system" with a "business application," and implies that the invention allows a "document management system," including the software application claimed therein, to coordinate with the existing "accounting system."

The '381 Patent Specification also refers to "Microsoft Office," Ex. 1001, col. 53, l. 38, whereas Patent Owner points to "Microsoft Word, Excel and Outlook," as examples of single applications. PO Resp. 19. The former example, Microsoft Office, like the "Internet," or "filing system," verifies that a software application is not limited to a single application. During the oral hearing, Patent Owner conceded that that "one of ordinary skill in the art would understand that [Microsoft Office is] . . . as an example as a destination that could include multiple discrete software applications," in response to a question by Judge Tierney. Tr. 26:16–19.

Although Patent Owner relies on its declarant, Mr. Weadock, prior testimony by Mr. Weadock in unrelated litigation, cited by Petitioner, supports Petitioner. For example, Mr. Weadock asserted in a declaration that "i[t] is difficult to define software products according to any specific grouping of files. Software products are typically defined according to their features." Ex. 1019, 2 ¶ 1. Mr. Weadock also declared that "[a]ttempting to define software as a particular collection of files is ultimately impossible if code units within the same file are shared. . . . Attempting to define software strictly as a collection of files is a fruitless exercise when some of those files perform double duty in different contexts." *Id.* at 6 ¶ 14. Quoting another source, he stated that "'[y]ou cannot isolate application code easily anymore.'" *Id.* (quoting "the director of information technologies for US Steel Group"). "'You know, it becomes very difficult to draw a specific line at which you've drawn a boundary between [an] operating system and application.'" *Id.* (quoting Scott Vesey, Boeing's Windows Web Browser Manager). "Indeed, both industry professionals and computer customers think of a software product more as that which enables a set of related features than as a collection of specific files." *Id.* at 6 ¶ 15.

Software, according to the '381 Patent Specification, is stored in "any multitude or combination of . . . storage devices." Ex. 1001, col. 62, ll. 43–46; Fig. 15 (depicting system components including memory devices 60, 62, 66, 68, 70, and processor/CPU 58). The software may reside on different servers and clients: "Alternatively, the engine object layer and the engine may be optionally located in a distributed environment on different machines, servers, and the like." *Id*. at col. 68, ll. 17–22. "The VC software can reside on a PC, LAN/WAN server, digital device (such as a digital copier) or on a web server to be accessed over the Internet." *Id*. at Abstract. The Abstract verifies that the software can reside on a device, and does not preclude firmware, as Patent Owner argues.

The title of the '381 Patent is "Distributed Computer Architecture and Process for Virtual Copying." The title bolsters the finding that the disclosed invention contemplates a software application that works in a distributed manner as a suite of programs, in different machines and on different memory locations, to accomplish various functions. The '381 Patent also discloses "combin[ing] with any . . . suitable processing circuits, including programmable logic devices, such as PALs (programmable array logic) and PLAs (programmable logic arrays), DSPs (digital signal processors) . . . ASIC's (application specific integrated circuits), VLSIs (very large scale integrated circuits) or the like." *Id*. at col. 63, ll. 10–16. This disclosure shows that the invention may include distributed firmware.

Accordingly, a "software application" is a program or group of programs which operate together in a system to perform a function or functions, and the programs can be stored in a variety of places on a variety of machines, and operate in a distributed manner. An application may include software and hardware and performs a function.

Third-party software application/external applications

Claims 1 and 14 recite "managing the electronic paper from at least one third-party software applications." In our Institution Decision, we determined that the terms "third-party software application" and "applications" mean "a program that may or may not be on a device." Dec. on Inst. 12. Patent Owner urges a narrower construction of a "third-party software application," and states that it means "a software application that is provided to the end user by a different manufacturer." PO Resp. 7. Patent Owner does not explain how a "different manufacturer" is involved in claim 1. Mr. Weadock urges that the term implies "commercially available software made by someone other than the creator of the computer data management system." Ex. 2002 ¶ 24.

The Specification refers to "third-party" software as "proprietary" software. *See* Ex. 1001, col. 8, l. 11. It also refers to "business applications (such as Microsoft Office, Microsoft Exchange, Lotus Notes)." *See id*. at col. 5, ll. 56–57; col. 46, ll. 19–21. Claims 1 and 14 are drawn to "a computer data management system," comprising an input module to manage a "third-party software application[]." Mr. Weadock and Patent Owner fail to explain how the maker of a software application has anything to do with the structure of computer data management system or the third-party software application. Claims 1 and 14 are system claims, not method claims. The '381 Patent specifically states that a "third-party" may create one of the modules of the claimed invention: "The Client Module can be a GUI that Imagination Software develops, or a third-party application that directly communicates with the Server Module." Ex. 1001, col. 80, ll. 47–49. This verifies that the identification of the programmer who creates any software that communicates with the claimed "computer data management system" does not structurally distinguish software made by the maker of the "computer data management system."

Similar to "applications," these related terms (third-party and external) do not preclude software that resides in printers, scanners, or other devices. An external application, according to the construction of "application" *supra*, does not preclude hardware, because it includes a file system and the Internet. As also discussed, the Specification refers to copying paper from "one device and[/]or application to another device and/or application," thereby further broadly blurring any distinction between a device and a device having a software application. Dec. on Inst. 12 (quoting Ex. 1001, col. 6, ll. 44–46). Further, software, according to the Specification, as explained above, is stored somewhere in the system, in "any multitude or combination of . . . storage devices." Ex. 1001, col. 62, ll. 43–46; Fig. 15 (depicting system components including memory devices 60, 62, 66, 68, 70, and processor/CPU 58).

Claim 1 requires a system that transmits electronic paper (i.e., "at least one of an electronic image, electronic graphics and electronic document") to "a plurality of external destinations including one or more of external devices and applications." This phrase informs that "external" means at a location remote from the source that transmits the electronic paper.

Accordingly, the term "third-party software application" means "programming code that may or may not be on a device." An "external application" means software, hardware, or both, at a remote destination, remote, for example, from the source that transmits electronic paper.

Module

Claim 1 recites a "computer data management system" comprising "at least one input module," "at least one module communicable with said at least one input, output, client, and process modules and external applications." In the Decision to Institute, we noted that one plain meaning of "module," is "a logically separable part of a program." Dec. to Inst. 13 (citing IEEE 100 THE AUTHORITATIVE DICTIONARY OF IEEE STANDARDS TERMS SEVENTH EDITION 704 (2000), *available at* http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4116801 (last visited Sept. 19, 2013)).

We determined that "each 'module' . . . is a logically separable part of the claimed data management system, and a module may include another module and overlap with another module in functionality." *Id.* at 14–15. Petitioner agrees with the definition, and Patent Owner does not. Pet. Reply 8; PO Resp. 6.

In reaching the construction, we also noted that the '381 Patent states that the modules have "counterparts" to "aspects" in conventional devices: "Each module [except possibly a server module] is a counterpart to an aspect that is found on a conventional copier." Dec. on Inst. 3 (quoting Ex. 1001, col. 71, l. 66–col. 72, l. 1). As to the server module, the '381 Patent Specification states that "[u]nlike conventional copiers, VC's Server Module is a unique *subsystem* that can communicate with the other modules." Ex. 1001, col. 72, ll. 44–47 (emphasis added). We further noted that "[t]he '381 Patent also states that '[t]he Client Module is generally simply an interface to the Server Module.'" Dec. on Inst. 13 (quoting Ex. 1001, col. 50, ll. 15–16). Therefore, a module in a software application may be a "unique subsystem" of that application, and as an "interface" to another module, may overlap with other modules. *See id.*

Notwithstanding these disclosures in the '381 Specification, Patent Owner maintains that the Specification provides no support for the determination that modules may include other modules and may overlap in functionality. PO Resp. 7. Patent Owner states that a "module" is "a logically separable part of the software application of the data management system that can function in a plug-and-play manner within a Virtual Copier." *Id.* at 8.

Patent Owner cites to its declarant, Mr. Weadock, as support. *Id.* (citing Ex. 2002 ¶¶ 26, 28–29). Mr. Weadock states that this definition constitutes a "more appropriate interpretation of 'module' in the context of the '381" Patent. Ex. 2002 ¶ 29. Mr. Weadock cites to the Institution Decision at page 10, where we noted that the '381 Patent states that "'[a]s long as the input and output [m]odule conform to the API specified in this document it will plug-and-play with VC.'" *See id* ¶ 27 (quoting the Institution Decision, which quotes Ex. 1001, col. 9, ll. 35–39).

Mr. Weadock's interpretation ignores the quoted qualifier in the Specification, which is missing from claim 1: "as long as the . . . Module conform[s] to the API specified in this document . . . it will plug-and-play." Although claim 10 recites a generic "application programmer interface (API)." No claim requires a module to conform to the "API specified," the "'C'-language API" or "COM-based interface," as specified in the '381 Patent Specification. *See* Ex. 1001, col. 50, ll. 21–34. None of the claims recite the "discrete" or "plug-and-play" feature.

Patent Owner chose not to limit the claims by qualifying the modules as "discrete" or "plug-and-play." The '381 Patent implies that a module, as set forth in the claims, is broader than any specific examples of discrete "plug-and-play" modules. In essence, a software module has boundaries defined by specific code that produces a specific software function associated in the claim with the module. This generic software module is "logically separable," because it can be defined by the logic code that produces its function, even if the module cannot be physically extracted from a single memory location as "plug-and-play" module.

Patent Owner also does not address broader implications in the '381 Patent Specification, including the module counterparts to prior art aspects, the modules as interfaces to each other, or the modules as subsystems. *See* Ex. 1001, col. 7, l. 67–col. 8, l. 1, col. 50, ll. 15–16, col. 71, l. 66–col. 72, l. 1, col. 72, ll. 44–47; *see* *also* Ex. 1005 ¶¶ 35–39 (Mr. Wibbels discussing the disclosed modules, including inconsistent descriptions in the ’381 Patent Specification of the server module, one of the five disclosed modules). The ’381 Patent does not describe these “counterparts” or “subsystems” as discrete modules, and an “interface” to another module implicitly overlaps in code and function with the module for which it interfaces. Therefore, the modules need not be discrete.

In the related ’302 IPR, during cross-examination by Patent Owner, Dr. Melen, petitioner (Ricoh’s) witness in that case, testifies that “the word ‘module’ is very broad and very nonspecific, and [can] be comprised of modules and modules of mod - - modules, modules spread across the network, modules which include other people’s code.” ’302 IPR, Ex. 2003, 144:16–20.

Dr. Melen similarly testified, when asked about the five modules claimed and disclosed in a related patent having the same Specification as the ’381 Patent, that

> I don’t think . . . module is necessarily one thing. You can have a module inside a . . . module. You can have a module which spans machines. Module is not so precise. *But what is more specific is exactly what they do. And so the question is, does [the prior art] talk about those basic functions of scanning and printing and - - yes. . . . It’s just software.*

Id. at 142:5–15 (emphasis added).

Dr. Melen’s testimony is not required to support this claim construction, but we employed the same claim construction in the ’302 IPR. His testimony informs our construction of the same term in the two proceedings—the ’381 Patent and the patent challenged in the ’302 IPR share a common specification.

In the ’302 IPR, Mr. Weadock candidly stated during cross-examination that a module may overlap in code with another module, retreating from statements in his declaration that may have been interpreted as absolutely precluding overlap in modules: "And normally when we talk about modules, we think of them as not overlapping, but there might be situations in which modules could share some code. There might be some common code between the two modules." '302 IPR Ex. 1013, 191:3–7. Mr. Weadock also acknowledged that separate functionality between modules may not be required: "I would hesitate to ever make any absolute statements when it comes to software. . . . Because there's so many different designers and so many different philosophies, but it would - - I can say that it would surprise me to see a modular software application with *heavy* overlap of functionality between the modules." *Id*. at 192:6–15. The experts, therefore, agree, that a module may share or overlap in code (which performs the function).

Petitioner points to another passage in the'381 Patent that implies that module functions overlap: "[W]hile the above discussion has separated the various functions into separate layers of functionality, the layers may be combined, physically and/or logically, and various functions may be combined together." Pet. Reply 8 (quoting Ex. 1001, col. 85, ll. 9–12).

Petitioner also points out that a book written by Mr. Weadock employs a definition of a "module" that is consistent with the meaning set forth in the Institution Decision: "a logical unit of separation in the application." Pet. Reply 4 (citing Ex. 1020, 1). Implicitly, this definition agrees with ours because code that performs a specific function in an application can be separated as a logical set of code that performs a required function.

According to the foregoing discussion, each "module," as recited in claim 1, does not require a discrete or plug-and-play feature, but each module is a logically separable part of the claimed "software application," demarcated by code corresponding to the specific function recited for that software module. Each software module may include another software module and overlap with another such module.

Claims 8 and 12, certain phrases

Claim 8 recites "wherein the one or more of the external devices and applications integrates the computer data management system into an external application via one of running the computer data management system, as an external service and embedding the computer data management system as an embedded service." Based on the claim language, "one of" refers to running and embedding, and means one of running or embedding. The claim does not define a relative internal system for the "external" application. Claim 8 depends from claim 1 and requires that the computer data management system includes a memory and a processor. It is not clear how that hardware portion of the claimed management system can be integrated by means of embedding or running. Accordingly, integrating by one of embedding and running, means that the computer data management system connects to, exists as, runs as, or was built as, part of another (i.e., external) application.

Patent Owner's declarant, Mr. Weadock, states that "the Board has a valid point" about the hardware not being embedded into anything, such as software, but "that a person of ordinary skill would understand that the 'wherein' clause of Claim 8 pertains to the software components of the computer data management system" as being embedded. *See* Ex. 2002 ¶ 36. Nevertheless, a reasonable interpretation, which does not require ignoring specific claim limitations that require the processor and memory hardware in the claimed management system to be "integrate[d] . . . into an external application," via "embedding . . . as an embedded service," implies that an external application includes hardware. The '381 Patent Specification supports this interpretation, for the reasons explained above. Another supported and reasonable interpretation is that "integrates" includes "running . . . as an external service," as claim 8 specifies, which implies that "integrates" means that two devices, systems, or applications, are compatible with each other.

Claim 12 recites "[a] computer data management system" that includes a "single function copy operation linking devices," "a one step programming method[,] . . . a method of recreating a module oriented copier in software," and "a copier interface." Claim 12 recites phrases that appear to place the claim into two statutory categories, "process" and "machine," which is impermissible under 35 U.S.C. § 101. For purposes of this proceeding, the "one step programming method" is interpreted either to describe functional characteristics of a machine, the data management system, or to include a method of making the "electronic business processes" in the machine, so that the machine includes "a module oriented copier in software," made "with no or minimal reprogramming." *See* Claim 12. This product-by-process step is not afforded patentable weight, because Patent Owner does not direct attention to a disclosure in the Specification that indicates that after the system has been made, it behaves differently than a system made with more than minimal reprogramming.

Claims 9–11

Claims 9–11 recite a "server module." Claim 9 depends from claim 7, which depends from claim 1, and recites "wherein the server module includes" the four means-plus-function limitations. However, "the server module" recited in claim 9 lacks antecedent basis, because claim 1 does not recite a server module. In our Institution Decision, we determined that "the server module" recited in claim 9 is interpreted either to refer back to "at least one module communicable," element

1.4, as recited in claim 1, or to refer to an additional module, a server module. Dec. on Inst. 17.

Patent Owner does not disagree with this interpretation. Rather, as noted above in the construction of "at least one of," during the oral hearing, Petitioner asserted and Patent Owner agreed, that claim 1 recites "at least one of" "at least one input module" or "at least one module communicable," in the alternative. Therefore, if claim 9 refers back to clause 1.4, "at least one module communicable," which claim 1 does not require necessarily because it is an alternative to clause 1.3, it follows that claim 9 does not limit claim 1 under this interpretation. Similarly, "the server module" recited in claims 10 and 11 also lacks antecedent basis, because they each refer back to claim 7, which refers back to claim 1. Accordingly, claims 9–11 recite limitations that do not limit the claims under the alternative interpretation that we employ, wherein "the server module" implicitly refers back to clause 1.4, "at least one module communicable," as recited in claim.

The Demand Letter

To support its claim construction, Petitioner submitted a demand letter (Ex. 1016, 26) by Patent Owner to accused infringers of Patent Owner's patents. *See* Pet. Reply 4 (arguing "it would be improper to ignore [Patent Owner's] prior statements for purposes of BRI [broadest reasonable interpretation].") The demand letter is not necessary to the claim construction or holding, but it serves to corroborate the claim construction of module and application outlined above. The demand letter implies that Patent Owner's claim construction of these terms corresponds to the constructions outlined above. [3]

[3] Patent Owner was questioned about the demand letter during the '302 IPR hearing and the '309 IPR hearing. *See* Tr. 18–22, IPR '302, Tr. 43.

For example, Patent Owner's demand letter states that

> [a] good example of an infringing system, and one your company likely uses, is an office local area network ("LAN") which is in communication with a server, employee computers having email software such as Outlook or Lotus, and a third-party scanner (or a multi-function printer with scanning functionality) which permits the scanning of a document directly to [an] employee email address as a pdf attachment. Such a system would be a typical example of what infringes. There are other examples listed further below.
>
> . . . [Y]ou may find it useful to consider, as illustrative examples, claims 1–5 of the '426 Patent. Reviewing those you can see that the patent claims are directed to a system having a digital copier/scanner/multifunction device with an interface to office equipment (or to the web) and related software, for scanning or copying and transmitting images electronically to one or more destinations such as email, applications or other local files. Coverage of this type of system, and of the more generally worded example [above] . . . , is further reflected in claims 1, 8 and 15 of the '410 Patent, claims 12 and 15 of the '381 Patent, and claims 9 and 16 of the '590 Patent.

Ex. 1016, 27.

Claim 5 of the '426 Patent, the subject of the related '302 IPR, includes "a software application" that "comprises . . . at least one input module . . . at least one output module . . . at least one process module . . . at least one client module . . . at least one client module . . . and at least one server module." *See* '302 IPR, Ex. 1001, col. 86, ll. 21–28. The demand letter implies that claim 5, which recites modules, similar to claim 1 of the '381 Patent, does not require discrete modules or a single application, because the claimed system is "directed to a system having a copier/scanner/multifunction device with an *interface to office equipment (or to the web) and related software*, for scanning or copying and transmitting images electronically to one or more destinations such as email, applications or other local files." Ex. 1016, 27 (emphasis added); *see also* Pet. Reply 4–5 (describing as "inconsisten[t]" Patent Owner's claim constructions in this proceeding and with respect to the demand letter).

Claim 12 of the '381 Patent recites "a computer data management system . . . wherein the system comprises . . . a method of recreating a module oriented copier in software." Claim 12 also recites "a copier interface implemented as [a] software application." The demand letter indicates that typical office systems that scan and transmit images "to one or more destinations" infringe claim 12. Therefore, in line with the claim construction, the demand letter and claim 12 imply that a software module need not be discrete and a "software application" need not be a single application.

B. Mr. Wibbels

Patent Owner asserts that Mr. Wibbels is not a person of ordinary skill in the art so that his opinions included in his declaration "should be given little to no weight." PO Resp. 12. Patent Owner contends that Mr. Wibbels's "experience with printers . . . is limited to firmware rather than high level software applications." *Id.* at 11. Patent Owner acknowledges that Mr. Wibbels has a B.S. in physics, a B.A. in mathematics, and an M.S. in mechanical engineering and "has had some formal instruction in the C language and is self-taught in the QuickBasis language." *Id.* at 10. According to Patent Owner, Mr. Wibbels also "worked on a number of printer hardware level algorithms, such as half-toning algorithms, calibration algorithms, and under-exposure algorithms." *Id.* at 10–11.

Patent Owner's assertions bolster the record evidence and show that Mr. Wibbels has experience and education in software. Patent Owner does not qualify "high level" software. Patent Owner had the opportunity to challenge the declaration of Mr. Wibbels through cross-examination and competing testimony.

See In re TMI Litig., 193 F.3d 613, 692 (3d Cir. 1999) ("So long as the expert's testimony rests upon 'good grounds,' it should be tested by the adversary process—competing expert testimony and active cross-examination." (quoting *Ruiz-Troche v. Pepsi Cola of P. R. Bottling Co.*, 161 F.3d 77, 85 (1st Cir.1998))). In addition to having a Master's degree in engineering, Mr. Wibbels has been a "Distinguished Technologist/Strategist" at Hewlett Packard Company since 2004. Mr. Wibbels is "an inventor of 17 issued U.S. Patents" and has "spent approximately 20 years working on product development of HP scanning and printing solutions." Ex. 1012, 1–2; Ex. 1005, 2.

The record shows that Mr. Wibbels possesses "at least . . . 'ordinary skill in the art'" (Ex. 1005 ¶ 20):

> the level of ordinary skill in the art of the '381 [Patent] at the time of the effective filing date is a person with a bachelor of science degree and two years experience in research or development (e.g., engineering, product development, requirements analysis) in fields or industries pertinent to the art (e.g., digital imaging systems, such as scanners or printers). With more education, for example post-graduate degrees and/or study, less industry experience is needed to attain an ordinary level of skill.

Id. ¶ 19.

The claims of the '381 Patent generally recite systems for processing and transmitting electronic images with "counterparts" in conventional printing systems. The '381 Patent Specification also summarizes the disclosed invention as involving C-level software. *See* Ex. 1001, col. 3, ll. 35–67. Firmware skill, contrary to Patent Owner's arguments, skill in creating computer algorithms, and other languages, including C-level, represent relevant skill. Mr. Weadock testifies that the requisite level of skill incudes experience in "printing, networking, scanning, and e-mail." Ex. 2002 ¶ 15. Patent Owner acknowledges that

Mr. Wibbels had "formal education" in software languages including C-level. PO Resp. 10. The record, including the prior art, which is directed to similar systems as the claimed invention, show that Mr. Wibbels has the requisite engineering education, and experience at least in imaging art, encompassing printer, scanning, and networking—"digital imaging systems." Ex. 1005 ¶ 19. Mr. Wibbels invented systems as specified in several relevant patents generally involving computer based imaging. *See* Ex. 1012, 1–2. Patent Owner has not demonstrated through cross-examination or otherwise that Mr. Wibbels's testimony is unreliable. Mr. Wibbels qualifies to aid in the Board's understanding and his opinion is entitled to due weight.

C. Asserted Grounds of Unpatentability

1. Cotte – Anticipation, Claims 1–15

<u>Cotte–Overview</u>

Cotte generally describes an integrated system that produces scanned document data that can be sent to a host computer and e-mailed, faxed, or printed. Ex. 1011, Abstract, Fig. 10, col. 10, ll. 42–58. Cotte's paper input device

> senses the insertion of a document to be scanned, initiates a host computer process, i.e., controls the host process by insertion of the paper and symbols on the paper, scans the images and text on the paper, . . . send[s] the scanned data to the host for further electronic processing such as display, transmission, storage or modification. Principally, this new technology is a paper input device using scanning technology which controls the host computer rather than the other way around.

Id. at col. 2, ll. 42–51.

Cotte also discloses that

> the user can put the document in the paper input device and the input device software will automatically scan the document, send the data to the host in any of the ways described herein, and the input device software resident on the host will then cause a pop-up window to

> appear on the screen where the image of the scanned document appears.

Id. at col. 16, ll. 58–63.

One way to send data involves symbol recognition software: "the input device includes symbol recognition software that can recognize symbols on the document to be scanned which indicate where the document is to be FAX'ed, sent as E-mail[,] etc." *Id*. at col. 11, ll. 27–31.

<u>Claim 1</u>

Relying on Mr. Wibbels's declaration (Ex. 1005), Petitioner reads the elements of claim 1 onto Cotte's scanning network system. Pet. 43–50. According to the '381 Patent, "VC can be viewed as a copier"—a copier in the sense of copying electronically—even though "VC does not distinguish between electronic and physical paper." Ex. 1001, col. 71, ll. 62–65. Based on the foregoing descriptions, the two systems are similar, in terms of the functionality as a basic copier operating on electronic and physical paper.

The preamble of claim 1 follows:

> A computer data management system including at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to . . . one or more of external devices and applications . . . connectable at least one of locally and via the Internet.

Cotte discloses the preamble, according to the discussion *supra* and Petitioner's showing. For example, as described *supra*, Cotte's input device system manages the host computer and sends an image and text to an e-mail, fax, printer, or other similar applications or devices that have software, thus showing that Cotte's system is "connectable at least one of locally and via the Internet," as the claim 1 preamble recites. *See* Pet. 43; Ex. 1011, Fig. 17, col. 10, ll. 35–58, col. 16, ll. 20–22 (disclosing local or wide area network); Ex. 1005 ¶ 291.

Claim 1 also recites the following limitations: "at least one processor responsively connectable to said at least one memory, and implementing the plurality of interface protocols as a software application" and "at least one memory storing a plurality of interface protocols for interfacing and communicating." Cotte discloses microprocessor 352 implementing software for controlling operations of the input device that is stored in ROM/RAM 132. *See* Ex. 1011, Fig. 11A. As Petitioner and Mr. Wibbels explain, Cotte's processor 352, which implements this stored software for fax and input communications, satisfies the listed limitations. *See* Pet. 43; Ex. 1005 ¶ 293; Ex. 1011, col. 8, ll. 37–55, col. 9, ll. 4–12. For example, "the input device software controlling the[ ]microprocessor 352 includes software routines to send the appropriate commands to the Fax modem to control its operations in sending the scanned data as a Fax." Ex. 1011, col. 9, ll. 4–12. ROM/RAM memory 132 stores the input device software. *Id*. at col. 8, ll. 37–39.

Claim 1 also recites

> at least one input module managing data comprising at least one of paper and electronic paper input . . . , and managing at least one imaging device to input the data through at least one of a scanner and a digital copier, and managing the electronic paper from at least one third-party software applications.

Claim 1 allows for each of the "managing" functions to be handled by one or more of the "at least one input module." To satisfy the input module limitations, Petitioner relies on Cotte's scanner, input device 114 at Figure 11A, which includes the stored fax protocol discussed *supra*. Pet. 43; Ex. 1005 ¶ 298. As noted, in Cotte, the "input device software controlling the[ ]microprocessor 352 includes software routines to send the appropriate commands to the Fax modem to control its operations in sending the scanned data as a Fax." Ex. 1011, col. 9, ll. 8–12. Cotte's input software also sends data automatically to a laser printer, thereby managing at least one imaging device to input data through a digital copier. *Id*. at col. 10, ll. 45–50, col. 11, ll. 33–37. Therefore, Cotte's input device software routines manage paper and electronic paper, through the scanner, an imaging device, thereby satisfying at least the first two recited functions of the input module.

Cotte also implies that input software routines automatically control data between the host and the fax modem, in both directions. *See* Ex. 1011, col. 8, ll. 39–56. Controlling this fax data transfer also satisfies the first, second, and third input module functions, the third being "managing the electronic paper from at least one third-party software application[]." As an example, microprocessor 352 programs UARTs 135 and 136 to receive fax data and send it to host CPU 110. Ex. 1011, col. 8, ll. 52–56, Fig. 11A. As another third-party software application, Cotte also discloses that the "input device includes symbol recognition software that can recognize symbols on the document to be scanned which indicate whether the document is to be FAX'ed, sent as E-mail[,] etc." Ex. 1011, col. 11, ll. 27–31. The claimed input module functions read on that process, because the symbol recognition software helps to manage electronic paper from a scanner using symbol software, which constitutes a third-party application. Software resident on the host may also generate a drop-down menu, which also constitutes a third-party application relative to the input device software. *Id*. at col. 10, ll. 53–57, col. 15, ll. 35–44. Also, sending data to a word processor after performing optical character recognition constitutes managing electronic paper from at least one third-party software application (the OCR third-party application). *Id*. at col. 10, ll. 53–57.

Patent Owner asserts that "Cotte's 'software routines,' that are implemented by a microprocessor 252 of an input device 114, do not anticipate a software ***application*** that comprises an input module, as required by claim 1." PO Resp. 13. Patent Owner asserts Cotte's microprocessor "appears to be nothing more than a hardware device (e.g., microprocessor 52) controlled by firmware." *Id.*

Patent Owner's arguments are not persuasive. According to the claim construction outlined above, a "software application," as set forth in claim 1, can include software stored in different places, including as firmware, which in this case includes RAM/ROM memory 132 in Cotte. *See* Ex. 1011, Fig. 11A, Fig. 17. The '381 Patent Specification also specifically mentions that the disclosed program can "instruct the central processing unit," Ex. 1001, col. 62, l. 39, and be stored in "ROM 60 and/or RAM 62" or another "memory media . . . [as] program information for controlling the computer to enable the computer to perform the functions described herein," *id*. at ll. 30–37.

Patent Owner also asserts that Cotte's "symbol recognition software cannot be an input module because the document has already been received"—"symbols cannot be recognized by software until after the document is scanned." PO Resp. 14. This argument does not address the alternative of the input module controlling the programmed UARTs and performing other scanner input functions, for example, inputs to OCR, OCR outputs to other software, or input software deciding to send the scan as input to the fax, as input to the host for further processing, as input to a RAM buffer, or as input to a printer (i.e., "managing data through at least one of a . . . printer"). *See* Pet. 43; Ex. 1011, col. 9, ll. 1–12. Further, "special software in the input device compresses the scanned data, sends the scanned data to the host and automatically carries out the desired processing" by receiving a menu command or symbol recognition command. Ex. 1011, col. 2, ll. 55–62. Therefore, even if claim 1 requires controlling the scanning, Cotte discloses compressing during scanning. Further, according to the '381 Patent Specification, as noted above, input modules have counterparts to prior art scanners. Cotte's input device software includes software to control the scan as a typical prior art input scanning device.

Patent Owner also argues that Cotte's system does not manage paper from a third-party software application. PO Resp. 14. Patent Owner describes as "problematic" the Board's finding in which Cotte's symbol software recognition software is an input module that both manages one imaging device and manages electronic paper from at least one third-party software application. *Id*. at 15. According to Patent Owner, "it would be illogical for a software application to receive electronic paper from itself." *Id.* at 15.

This argument is not clear, because the recited software application communicates with the recited third-party application. Cotte includes several third-party applications in addition to the software recognition software, such as OCR, e-mail, or fax. The argument also conflates the symbols, or third-party software which creates the symbols, with the recognition software in the input device or the host, the input module. The symbols, placed on the paper by a third-party software application, or the symbols as a third-party software application, are processed by the recognition software, or input module, which recognizes those symbols. For example, Cotte states that "the recognition software [input module] is taught to recognize the bit map of sticker 400 [the third-party software application] as a command to invoke the appropriate host software to receive the scanned image of document 402 and send it to the FAX modem 118 in Fig. 10."

Ex. 1011, col. 12, ll. 11–14. "[T]he symbols may be printed on the document using [third-party application] software . . . ." *Id.* at col. 11, ll. 34–35.

As Patent Owner recognizes, Cotte states that the recognition software can be in the form of "specific 'macros' or predefined sequences of instructions." *Id.* at col. 11, 48–49; *see* PO Resp. 20. Although Patent Owner's declarant, Mr. Weadock, concludes that a "macro" is not a "module," Mr. Weadock fails to set forth a meaningful distinction, and refers to a "macro" as a "predefined script of commands to be executed as a unit." Ex. 2002 ¶ 77. These macros fit Patent Owner's definition of a "discrete" set of instructions of a module, and the broader construction outlined above wherein the functions define the software module.

Patent Owner's similar argument that "Petitioner seems to confuse a software function with a software module" relies on an assertion that a module requires "'a consistent API [application programmable interface].'" PO Resp. 19 (quoting Ex. 2002 ¶ 77). The recited modules, however, do not require an API.

Patent Owner also alleges that Cotte's system does not implement a plurality of software protocols as a software application to interface and communicate with a plurality of external destinations. *Id.* at 16. Patent Owner also asserts that a fax protocol and printer driver do not constitute a plurality of protocols. *Id.* Patent Owner acknowledges that Cotte's system "communicates with other devices," but argues that the evidence does not show that "'a plurality of protocols' is in fact implemented as a software application." *Id.* at 17.

These arguments are not persuasive. Mr. Wibbels testifies that Cotte discloses transmitting data using protocols, including printer drivers and fax protocol software. *See* Ex. 1005 ¶ 293. Mr. Weadock does not rebut this testimony persuasively. In response to the assertion that a printer driver constitutes an additional protocol to a fax protocol, Patent Owner responds that a scanner would not have its own "printer driver that is implemented as a software application." PO Resp. 16. Based on the claim construction, claim 1 does not preclude the location of the printer driver software from being stored in distributed manner acting with the recited software application.

Patent Owner acknowledges that a printer driver "is a discrete software program." *Id.* The record shows that several protocols exist necessarily to communicate with different devices and their software, including a scanner, an e-mail service, a printer, a Clipboard, or a fax. *See, e.g.*, Ex. 1011, Fig. 17, col. 14, ll. 28–30; Pet. Reply 5. For example, Cotte supports this finding, by disclosing the mapping of an e-mail address and "communication protocol" for the e-mail. *See* Ex. 1011, col. 13, ll. 4–6. Cotte also discloses "any form of transmission of data to the host including transmission over a local or wide area network, satellite, packet radio, ISDN transmission, etc." *Id.* at col. 16, ll. 20–23. Cotte discloses "prior art image compression algorithms" and "[a]ny known data compression process." *Id.* at col. 14, ll. 56–60. Cotte discloses "parallel format transmission protocol," *id.* at col. 14, l. 67–col. 15, l. 1, or a "serial protocol," such as RS232, or a fiber optic link, *id.* at col. 17, l. 40–50. These different types of transmission and compression formats constitute different protocols. Petitioner points out that Patent Owner asserted in previous litigation, asserting infringement of a similar limitation in a related patent, that using "'protocols is a *requirement* . . . to communicate and interface with external devices and applications.'" Pet. Reply 5 (quoting Ex. 1022 ¶ 151, emphasis by Petitioner).

Claim 1 also recites "at least one module communicable with said at least one input, output, client, and process modules and external applications, and capable of dynamically combining the external applications with at least one of digital capturing devices and digital imaging devices." As noted in the Claim

Construction section, the parties agree that this clause (1.4) is alternative to the preceding clause, the "at least one input module" (1.3). It follows that Cotte need not disclose both clauses to anticipate claim 1.

Nevertheless, in the alternative, Petitioner maintains that Cotte discloses the following: "A module in communication with an input module. Upon receiving a scanned image, other modules attach a scanned image to a clipboard, an email, or send it as a fax." Pet. 43 (citing Ex. 1011, col. 10, ll. 43–53, FIG. 17; Ex. 1005 ¶ 295).

At the cited passage, Cotte discloses host software, which includes

> a drop down menu 250 presenting options to the user regarding what should be done with the scanned image. These menu options can be such things as "FAX this image" as symbolized by icon 253 or "Send this image as an E-mail message" as symbolized by icon 255, or "Send this image to the laser printer for printing" as symbolized by icon 257 . . . .

Ex. 1011, col. 10, ll. 43–50; *see also id.* at Fig. 17, col. 4, ll. 46–49 (describing a typical pop-up window based on software which exists on the host machine).

In other embodiments, Cotte describes performing similar processes using codes in an input stream, or using the symbols described above, using automatic processing, "without the user having to press any buttons, make any menu selections." *Id*. at col. 11, ll. 42–43. Cotte implies that both sets of software can be used together, because the input device "includes" recognition software, and the separate host computer includes the typical pop-up menu. *Id*. at col. 11, l. 28, col. 10, ll. 44–48. The description of "without the user having to press any buttons, make any menu selections," implies the user may make selections if the user runs out of stickers having symbols through the existing host pop-up menu. *See* col. 11, l. 44 – col. 12, l. 4, Fig. 17. In the preferred embodiment, the input device "includes symbol recognition software." *Id*. at col. 11, l. 27. Therefore, Cotte generally implies that the existing host computer retains its pop-up menu in some embodiments that include symbol recognition software as part of the input device.

Patent Owner also argues that Cotte does not disclose the element of "capable of dynamically combining the external applications with at least one of digital capturing devices and digital imaging devices." PO Resp. 20–21 (emphasis omitted). One or more of Cotte's symbols, software for creating same, the e-mail software, fax software, or word processing comprise external applications, which are capable of being combined with "at least one of digital capturing devices and digital imaging devices." These host packages are external to the input device, for example. *See* Ex. 1001, col. 10, ll. 29–58 (disclosing sending an image as an e-mail or fax, or performing word processing on it).

Cotte's drop down menu or symbol recognition software each constitutes "at least one module communicable with said at least one input . . . modules," with input modules including the UART input module software, or the input module software that controls scanning functions, etc., as identified *supra*. *See* Pet. 43 (referring to other modules that attach a scanned image to an e-mail, clipboard, or fax). The menu software or recognition software is "capable of dynamically combining" the external fax application with at least digital capturing devices or digital imaging devices, such as, for example, the scanner or printer. In addition, similar to e-mail or fax, OCR or word processing each constitutes an external application relative to the input device, and those applications can be combined dynamically with "at least one of digital capturing devices and digital imaging devices" by processing data from those devices.

The host software communicates with the input device software, and includes the "appropriate software package," including the "host FAX software," or "E-mail software." Ex. 1011, col. 10, l. 67, col. 12, l. 32, col. 13, ll. 35–37, 50–

54, col. 15, ll. 55–64, col. 17, ll. 51–61, col. 20, ll. 60–63. The host packages include word processing and OCR described *supra*. The input device includes a "recognition software portion." *Id.* at col. 13, l. 13. The fax, e-mail, word processing, printing, OCR software, file storage, and other packages reasonably constitute or include "at least one input, output, client, and process modules and external applications," with "at least one module communicable" also reading on part of the input device recognition software or the host menu software, each of which may generate commands to invoke the appropriate host software. Ex. 1011, col. 13, ll. 36–49, 14, ll. 30–34; Ex. 1005 ¶ 295. For example, the display portion of the host menu itself reasonably constitutes a client module, and OCR, word processing, or compression software, reasonably constitutes a process module.

Cotte's input device software "invoke[s] specific types of software commonly found on user's computers or to invoke specific 'macros' or predefined sequences of instructions." Ex. 1011, col. 11, ll. 47–49. Therefore, similar to the explanation above, either the menu software, the part of the input device software that invokes one of the macros, the input software on the host that responds to the input device invoking commands, or one or more of the macros represent, "at least one module communicable with said at least one input" module. *See* Dec. on Inst. 25. Patent Owner responds by stating that the macros replace the recognition software, but it is not clear how that shows a missing limitation. *See* PO Resp. 20. Both constitute a form of symbol recognition software that may employ stickers (part of the third-party software discussed above). *See* Ex. 1011, col. 11, ll. 44–66. Patent Owner also argues that "[t]he Petition does not specify how Cotte discloses a macro or symbol recognition software that communicates with an input module, an output module, a client module, a process module, ***and*** external applications." PO Resp. 20. This argument is not clear, but it appears to ignore the limitation "at least one" in claim 1, i.e., "communicable with at said least one" input module and external application. As discussed in the Claim Construction section, the parties agree that "at least one" only specifies alternatives. In any event, as explained in the preceding paragraph and above, Cotte discloses module software such as e-mail, fax, UART control, printer drivers, OCR, word processing, and compression algorithms, thereby disclosing one or more of the claimed modules.

Patent Owner argues that Petitioner's showing improperly combines disparate software elements as a "<u>single</u> software application" and covers "firmware." *See* PO Resp. 19, 18–19. This argument is not persuasive because it assumes an overly narrow construction of "software application."

Patent Owner contends that Cotte requires that a command be manually entered, so there is no disclosure of "dynamically combining." *See* PO Resp. 21. This argument is not persuasive. Cotte's system automatically combines actions through the symbol recognition software or menu options. Claim 1 does not specify when any dynamic action starts, and Patent Owner does not clearly demarcate a distinction based on the claim or even the disclosure (assuming arguendo a limiting disclosure). *See id.* at 20–21. Pressing the button in Cotte causes a dynamic action because after it is pressed, the system responds with an e-mail, fax, etc. In the '381 Patent, a user similarly employs a "GO" command, according to claim 5, for example.

Based on the foregoing discussion and record, Petitioner shows by a preponderance of evidence that Cotte anticipates claim 1.

<u>Claim 2</u>

Patent Owner argues that "Cotte fails to disclose one or more of the external devices and applications include a printer, a facsimile, **and a scanner**." PO Resp. 22. Patent Owner's argument does not explain clearly why the printer, facsimile, and scanner of Cotte, identified above, and in the Institution Decision, do not anticipate claim 2.

Claim 1 requires the system to be capable of sending one or more of electronic images, documents, or graphics to "external destinations." These "external destinations" include "the one or more of the external devices and applications" referenced in claim 2. However, Patent Owner does not maintain that Cotte's system must send images, documents, or graphics *to a scanner* at one of such destinations to anticipate claim 1. Patent Owner does not contend specifically that the '381 Specification provides support for sending electronic images, documents, or graphics, from the system to a scanner. Rather, during oral argument, Patent Owner raised an untimely new argument, contending that a scanner and printer are part of the same "multifunctional peripheral" device, and that claim 2 requires sending an electronic document to such a multifunctional device. That argument, presented for the first time at the oral hearing, is waived. *Compare* Tr. 39:10–20, *with* PO Resp. 22.

Even if the argument is not waived, claim 2 does not explicitly require a scanner and printer to be at a single "external destination[]." Claim 2 refers to an "external destination," in claim 1, but not a single "external destination." Claim 1 recites "transmitting . . . to a *plurality* of external destinations including *one or more* of external devices and applications." Each one of the plurality may include one or more external devices. Claim 2 requires "one or more of the external devices" to include "a printer, a facsimile, and a printer." Hence, one external device may be a printer, another may be a facsimile, and another may be a scanner. Claim 1 requires transmitting to a plurality (e.g., two) of the devices, for example, a printer and a facsimile. Claim 2 does not require the capability to transmit to all the external devices, including the one that is a scanner. Therefore, claim 2 reads on Cotte's system, because Cotte's system has the capability to transmit to at least two of three external devices, a printer, a facsimile, and a scanner, as claim 1 requires, and one or more of the external devices include a printer, a facsimile, and a scanner, as claim 2 requires.

Based on the foregoing discussion and record, Petitioner shows by a preponderance of evidence that Cotte anticipates claim 2.

Claim 5

Claim 5 recites an interface that enables copying images between physical devices, applications, and the Internet using a single "GO" operation. Petitioner generally relies on Cotte's drop down menu, which provides a list of options, as explained above in connection with claim 1. *See* Pet. 44. Patent Owner argues that Cotte does not disclose transfers over an Internet, because Cotte discloses RS232 communications. PO Resp. 23. Patent Owner also argues that Cotte's "menu is directed to images that have ***already been scanned***," so it does not disclose "a single 'GO' operation." *Id.*

Patent Owner's assertion of a lack of Internet capability is not persuasive. Cotte discloses "any form of transmission . . . over a local or wide area network, satellite, packet radio, ISDN transmission, etc." *See* Ex. 1011, col. 16, ll. 20–23. Mr. Wibbels supports Petitioner, declaring that "sending an email message would include use of either a local area network or the Internet." Ex. 1005 ¶ 291. Mr. Weadock does not rebut this, but instead, states that in the 1992 timeframe, the Internet was not well-known. Ex. 2002 ¶ 72. Petitioner points out that 1992 is not the relevant timeframe. *See* Pet. Reply 6. Petitioner's point is persuasive, based on the earliest possible effective filing date for the '381 Patent of November 13, 1998. Therefore, skilled artisans would have recognized that Cotte discloses the transfer of images over the Internet at the time of the invention.

Patent Owner's argument that a single "GO" operation does not apply to an image that has "already been scanned" also is not persuasive. Patent Owner explains that "copying" cannot occur because scanning occurred first. *See* PO Resp. 23. Contrary to the argument, claim 5 requires "copying images between physical devices, applications, and the Internet." It is not clear what "copying . . . between . . . the Internet" means. In any event, claim 5 does not preclude "copying images between" a scanned image at a scanner and another device. In other words, "copying . . . between" as set forth in claim 5, reasonably means repeating or transmitting electronic image data. Cotte discloses automatically copying a scanned image or document to any software package, or internal or external storage devices, using a pop-up menu. *See* Ex. 1011, col. 18, ll. 54–56, Fig. 17.

Alternatively, as explained in the Institution Decision, "Figure 22 similarly discloses a copy button 310. Pressing the button causes the system to scan the image and send it to a printer for copying." Dec. on Inst. 26 (citing Ex. 1011, col. 19, ll. 42–48). Therefore, Cotte also discloses copying images and sending them to a laser printer by "pressing copy button 310" (i.e., a single "GO" operation) before the scanner scans the images. *See* Ex. 1011, col. 19, ll. 42, 36–55. Patent Owner does not challenge this finding. Based on the foregoing discussion and record, Petitioner shows by a preponderance of evidence that Cotte anticipates claim 5.

Claim 7

Claim 7 depends from claim 1 and recites "wherein the software application comprises at least one output module managing the data output from the computer data management system," as set forth in claim 1. The Institution Decision notes Petitioner's citation to Cotte's blocks 328, 326 in Figure 23 as representing the claimed output module, as follows: "Cotte's software that manages data," and block 344, which checks scripts, in Figure 24. Dec. on Inst. 27–28 (citing Ex.

1011, col. 8, ll. 38–52, col. 10, ll. 49–50, col. 18, ll. 52–55). Block 328 ("SEND IMAGE DATA TO LASER PRINTER") is part of "a flow chart of processing by input device software resident in both the paper input device and the host to implement a photocopy option." *Id.* at col. 5, ll. 3–6. Block 344 also represents part of a "flow chart of the processing" that employs "HOST RESIDENT SOFTWARE." *See id.* at col. 5, l. 8, Fig. 24.

Patent Owner argues that "it is possible for the printer [of Cotte] to be hardwired or to have firmware," so that it does not constitute "at least one output [software] module," as set forth in claim 1. PO Resp. 24. Patent Owner's argument about possible printer hardware in Cotte fails to address the cited software teachings. Further, Patent Owner acknowledges that a printer driver "is a discrete software program." *Id.* at 16.

Patent Owner also asserts that Cotte does not disclose a "modular software application," *id.* at 24, and makes a similar argument about Cotte's alleged failure to disclose "a discrete 'process module,'" *id.* at 25. Patent Owner also argues that Cotte does not disclose a "single software application," and cites "Microsoft Word," as an example thereof. *Id.* at 19. Patent Owner makes a similar argument that the claimed process module requires a discrete module that Cotte does not disclose. *See* PO Resp. 25 ("assuming, for the sake of argument" that converting grayscale pixels satisfies the process module function).

These arguments rely on overly narrow claim constructions that the record does not support. Claim 7 does not recite or require the modules to be "discrete." The record shows that Cotte's system performs the claimed output or process functions using software. It follows that the software code in Cotte that produces the function, under a broadest reasonable claim construction, constitutes a software application that comprises a module for that function.

Claim 7 also recites "at least one client module" presenting the paper or electronic paper data and "information related to at least one of the input and output functions." Claim 7 requires the client module to present the electronic paper data, and information related to at least one of the input and output functions. Petitioner cites Cotte's pop-up displays, which include a display of the incoming electronic paper data, as reading on these client modules. Pet. 45 (citing Ex. 1011, col. 18, ll. 50–55, col. 16, ll. 60–66).

Petitioner also cites Figure 17 of Cotte (entitled "Incoming Pages"). *See* Pet. Reply 12. Figure 17 displays a pop-up menu showing different options for incoming pages. *See* Ex. 1011, col. 16, ll. 60–66, col. 18, ll. 50–55. Displaying the pages constitutes presenting paper and electronic paper data, because the data reveals relative gray scale data values or document types, etc., while portraying written output options for archiving, photocopying, mailing, and faxing, constitutes providing the requisite input and output function information. *See* Ex. 1011, col. 17, ll. 5–10, Fig. 17.

Patent Owner also argues that Cotte does not disclose a client module presenting data "as it is being copied," as recited in claim 7. According to Patent Owner, Cotte's pop-up window does not anticipate claim 7, because Cotte's host may display the document after it is copied. *See* PO Resp. 25–26. The argument fails to address Figure 17 of Cotte, which includes pop-up windows with the title "Incoming Pages," shows pages under the title, and shows function buttons listed as "Photocopy," "Fax," and "Mail," among others. Cotte states that "incoming data is also displayed in a pop-up window." Ex. 1011, col. 18, ll. 52–53.

During oral argument, Patent Owner argued that claim 7 requires the top of a document to be displayed while the bottom of the document is being copied, and also indicated that if fifty pages are being scanned, then some of the first pages would appear during the scanning of the other pages. Tr. 42:22–43:2, 44:10–16. Patent Owner does not point the Board to support for this interpretation. In any event, the argument recognizes that an inherent delay exists between presenting and copying data in the claimed system. In Cotte's system, a user can change the gray scale after viewing that document on the computer screen, in order to obtain another view of another copy of the document in the new selected gray scale. This also implies, in line with the "Incoming Pages" title, that a document image is displayed at least while it is being copied, or copied a subsequent time. *See* Ex. 1011, col. 16, ll. 57–66.

Based on the foregoing discussion and record, Petitioner shows by a preponderance of evidence that Cotte anticipates claim 7.

<u>Claim 8</u>

Patent Owner argues that Cotte does not "disclose embedding any software as an embedded service." PO Resp. 26. Patent Owner points to VC as an embedded service in a third-party application. "'For example, rather than have VC as a separate application, a special button can be placed on a third-party application that launches VC in the background as illustrated in FIG. 33.'" *Id.* at 27 (quoting Ex. 1001, col. 73, ll. 43–49). Patent Owner contends that "Cotte's pop-up window is simply a user interface that is not an embedded service." *Id.* In the Institution Decision, we initially determined that "the computer management system software is integrated with an external application and external device and application by running part of it on the host and part of it on the external scanning device. *See* Ex. 1011, Fig. 11A. Petitioner sufficiently shows that Cotte anticipates claim 8." Dec. on Inst. 29.

Patent Owner does not rebut the findings. Claim 8 does not require an embedded service. Claim 8 recites two alternatives, "wherein the one or more of the external devices and applications integrates the computer data manage system into an external application via *one of* running the computer data management system, [1] *as an external service* and [2] embedding the computer data management system." Patent Owner does not dispute that Cotte's computer data management system is integrated with an external application, and runs as an external service, as set forth in the Institution Decision.

Moreover, as discussed in the Claim Construction section above, in the Institution Decision, and during the oral hearing, it is not clear how the memory and processor hardware of the computer data management system, as recited in claim 1, can be "embedded" as recited in claim 8. *See* Tr. 37:22–25; Dec. on Inst. 20. Patent Owner acknowledged during the oral argument that the hardware recited in claim 1 must be read out of claim 8 for "embedded" to have a clear meaning. *See* Tr. 37:22–25 (MR. GANTI: "Right, I think one of ordinary skill in the art would know that that's not a probable interpretation and it would be impossible to understand a system that has hardware and embed that into a system."). It is not clear how the hardware and software can be "integrated into an external application," unless the "external application" includes software and hardware. Cotte's system includes integrated software and hardware, reasonably constituting an external application, because it runs partly on the host and the scanning input device.

Based on the foregoing discussion and record, Petitioner shows by a preponderance of evidence that Cotte anticipates claim 8.

Claims 9–11

These claims depend from claim 7, which depends from claim 1, and further limit "the server module." As discussed in the Claim Construction section, and in the Institution Decision, we determined that "the server module" lacks clear antecedent basis and implicitly refers back to "at least one module communicable" in claim 1, under one claim interpretation. As also discussed in the Claim Construction section, claim 1 does not require "at least one module communicable," because it recites "at least one of" "at least one input module" and "at least one module communicable." As noted, Patent Owner agrees that both modules (*see supra* 1.3 and 1.4 of claim 1) are not required to satisfy claim 1, from which claims 7–9 depend. *See* PO Resp. 5 ("Patent Owner generally will not take issue with the Board except in the follow[ing] instances."); Tr. 31:6–9; Ex. 2002 ¶ 17.

Patent Owner's arguments regarding claims 9 and 10 are directed to functions recited with respect to the server module. PO Resp. 27–31. Patent Owner does not argue claim 11 separately. Under the broadest reasonable claim construction, claims 9–11 fail to limit claims 1 and 7, because "the server module," which refers under this construction to the "at least one module communicable," is claimed in the alternative to "at least one input module," in claim 1. Therefore, the recited dependent functions essentially describe alternative features that claims 9–11 do not require necessarily.

Based on the foregoing discussion and record, Petitioner shows by a preponderance of evidence that Cotte anticipates claims 9–11.

Claims 12 and 15

Claims 12 and 15 recite limitations that are similar to those addressed above in connection with claim 5. Patent Owner argues that Cotte does not disclose a "single function copy operation linking devices, applications and the internet" as recited in claims 12 and 15. PO Resp. 33 (emphasis omitted). Patent Owner argues that "because [Cotte's] menu is directed to images that have ***already been***

scanned, the menu cannot be intended for 'copying'." *Id*. These arguments are not persuasive.

Patent Owner does not explain how the recited "single function copy operation linking devices," as set forth in claims 12 and 15, precludes a menu directed to images that already have been scanned or requires something more. Cotte's menu provides for copying scanned images. As noted, a similar limitation is addressed above in connection with claim 5. For the reasons discussed above, Cotte discloses the copying as set forth in claims 5, 12, and 15. Similar to the discussion in connection with claim 5, claims 12 and 15 do not preclude electronic copying after scanning, and even if they do, Cotte discloses pressing a button to copy and send paper before scanning it. The demand letter, outlined above, asserts that many familiar systems that send scanned images to an e-mail or other software systems infringe claims 12 and 15. Cotte's system performs scanning and sending, in the same manner as the allegedly infringing systems described in the demand letter. The Petition and Institution Decision outline how the claims read on Cotte's system. *See* Pet. 49–51; Dec. on Inst. 32– 33 (discussing the Petition). During the oral hearing, Patent Owner contended that "[t]o the extent the Board will take that demand letter into account, it should only be done so with respect to . . . claims 12 and 15." Tr. 21:10–16.

Regarding the Internet limitation, Patent Owner maintains that Cotte copies an image from a scanner via a single serial port. PO Resp. 33 (citing Ex. 2002 ¶ 78). According to the discussion of claim 5, the record shows that skilled artisans at the time of the invention would have recognized that sending e-mails over wide area networks, or using satellite, packet radio, and ISDN transmission, as Cotte discloses, at least implies the Internet.

Based on the foregoing discussion and record, Petitioner shows by a preponderance of evidence that Cotte anticipates claims 12 and 15.

Claim 13

Claim 13 recites "a list of available module means." Citing Mr. Weadock's declaration, Patent Owner argues that Cotte does not disclose a list or a list that is read on startup. PO Resp. 30. During the oral hearing, Petitioner was asked "what's the name [in the list] of the process module and server module [in the prior art]?" *See* Tr. 52:1–2. Petitioner conceded "[t]here is no specific name." *Id.* at 52:23–53:5.

Petitioner does not show that Cotte discloses the claimed list as required by claim 13. In its Reply, Petitioner contends that "[c]laim 13 . . . only references other modules in a manner that lacks antecedent basis." Pet. Reply 7. Claim 13 references "said . . . modules," however, Petitioner's explanation falls short of explaining how a list of modules cannot be afforded patentable weight. In its Petition, Petitioner asserts that Cotte discloses a list of menu options "used at startup of a copier." Pet. 46 (citing Ex. 1011, col. 15, ll. 36–42, col. 18, ll. 43–51, col. 22, ll. 53–61, col. 23, ll. 19–25; Ex. 1005 ¶ 308). Nevertheless, Petitioner does not point out the specific list or show it is read on start-up. For example, the menu cited at column 15, lines 36–42, appears to be generated in response to an interrupt from the input device. *See* Ex. 1011, col. 15, ll. 30–32. Mr. Wibbels appears to rely on inherency "in a Windows implementation," Ex. 1005 ¶ 308, but Petitioner does not provide a citation or persuasive evidence that shows that Cotte employs a Windows implementation. *See* PO Resp. 31.

On this record, Petitioner does not show how Cotte discloses the claimed list of modules, or the list read on start-up. Petitioner thus does not show by a preponderance of evidence that Cotte anticipates claim 13.

Claim 14

Claim 14 is similar in scope to claim 1. Patent Owner groups its arguments regarding claims 1 and 14 together. *See* PO Resp. 12, 18, 20–21. Patent Owner's arguments are not persuasive, as explained above in connection with claim 1. Based on the foregoing discussion and record, Petitioner shows by a preponderance of evidence that Cotte anticipates claim 14.

2. SJ5

Based on the finding of anticipation by Cotte of claims 1–12, 14, and 15, it is not necessary to reach the ground of anticipation of these claims by SJ5. On the other hand, it is necessary to reach the ground of anticipation of claim 13 by SJ5.

Patent Owner makes a number of arguments regarding claim 13. PO Resp. 44–47. Petitioner does not respond to the arguments. For example, Patent Owner asserts that SJ5 does not disclose "maintain a list of available module means for maintaining a registry containing a list of said input, output, and process modules." PO Resp. 45 (emphasis omitted). Patent Owner also maintains that SJ5 does not disclose the claim phrase "maintain currently active modules means for maintaining said input, output, and process modules currently being used." *Id.* at 46 (emphasis omitted). Patent Owner also contends that a list of destinations is not the same thing as a "registry," and that SJ5 fails to show a registration process or even suggest that any software module is registered. *Id.*

Patent Owner also maintains that "SJ5 does not anticipate a server module that includes 'maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file,' as recited by claim[] . . . 13." *Id.* at 47 (emphasis omitted).

As noted above, at the oral hearing, Petitioner was unable to specify the relevant names for the process or server modules appearing in a list or lists recited. *See* Tr. 52–53. The Petition generally refers to the analysis of claims 1, 2, and 7, to address the limitations in claim 13. *See* Pet. 13. However, those claims recite different elements. Petitioner fails, among other things, to specify the required lists of modules. On this record, Petitioner does not specify clearly how SJ5 discloses all the recited and argued elements. Based on the foregoing discussion, Petitioner does not show by a preponderance of evidence that SJ5 anticipates claim 13.

III. CONCLUSION

Patent Owner does not present persuasive separate arguments for claims 3, 4, and 6. Patent Owner's remaining arguments track arguments addressed above and are not persuasive. Considering the record, including Petitioner's showing in the Petition and otherwise, and Patent Owner's arguments and evidence, Petitioner has demonstrated by a preponderance of evidence that Cotte anticipates claims 1–12, 14, and 15. Petitioner has not demonstrated by a preponderance of evidence that claim 13 is unpatentable.

IV. ORDER

In consideration of the foregoing, it is hereby

ORDERED that claims 1–12, 14, and 15 of U.S. Patent No. 6,771,381 B1 are unpatentable;

FURTHER ORDERED that, because this is a final decision, parties to the proceeding seeking judicial review of the decision must comply with the notice and service requirements of 37 C.F.R. § 90.2.

For Petitioner:

Stuart Meyer
Jennifer Bush
Fenwick & West LLP
Smeyer@fenwick.com
jbush@fenwick.com

For Patent Owner:

Vivek A. Ganti
Scott Horstemeyer
N. Andrew Crain
THOMAS | HORSTEMEYER, LLP
vg@hkw-law.com
scott.horstemeyer@thomashorstemeyer.com
andrew.crain@thomashorstemeyer.com

US006771381B1

(12) **United States Patent**
Klein

(10) **Patent No.: US 6,771,381 B1**
(45) **Date of Patent: Aug. 3, 2004**

(54) **DISTRIBUTED COMPUTER ARCHITECTURE AND PROCESS FOR VIRTUAL COPYING**

(76) Inventor: **Laurence C. Klein**, 1010 Wayne Ave., Silver Spring, MD (US) 20910

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/438,300**

(22) Filed: **Nov. 12, 1999**

Related U.S. Application Data

(60) Provisional application No. 60/108,798, filed on Nov. 13, 1998.

(51) **Int. Cl.**^{7} **G06K 15/00**
(52) **U.S. Cl.** **358/1.15**; 358/1.1
(58) **Field of Search** 358/1.1, 1.6, 1.13, 358/1.15, 1.16, 402, 403, 407, 425; 710/8, 14, 15, 33, 62, 63, 64, 65, 72, 73

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,666,495 A * 9/1997 Yeh 710/303

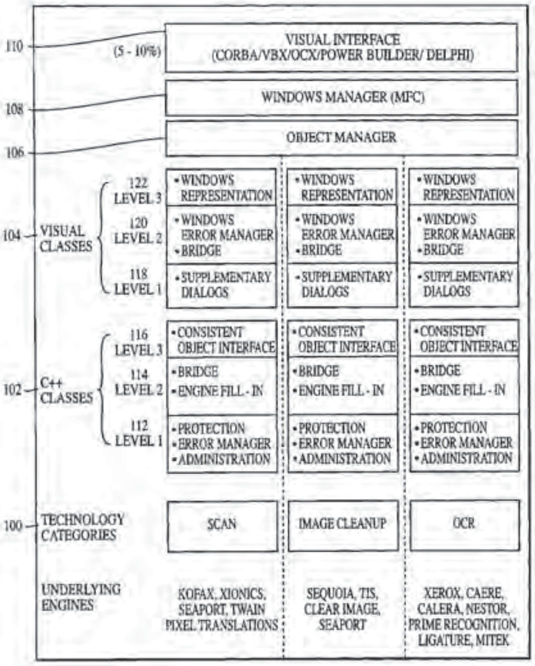
* cited by examiner

Primary Examiner—Arthur G. Evans
(74) *Attorney, Agent, or Firm*—Irah H. Donner, Esq.; Wilmer, Cutter, Pickering Hale and Dorr LLP

(57) **ABSTRACT**

The purpose of the Virtual Copier invention ("VC") is to enable a typical PC user to add electronic paper processing to their existing business process. VC is an extension of the concept we understand as copying. In its simplest form it extends the notion of copying from a process that involves paper going through a conventional copier device, to a process that involves paper being scanned from a device at one location and copied to a device at another location. In its more sophisticated form, VC can copy paper from a device at one location directly into a business application residing on a network or on the Internet, or visa versa. The VC invention is software that manages paper so that it can be electronically and seamlessly copied in and out of devices and business applications (such as Microsoft Office, Microsoft Exchange, Lotus Notes) with an optional single-step Go operation. The VC software can reside on a PC, LAN/WAN server, digital device (such as a digital copier), or on a web server to be accessed over the Internet.

15 Claims, 44 Drawing Sheets



HP 1001

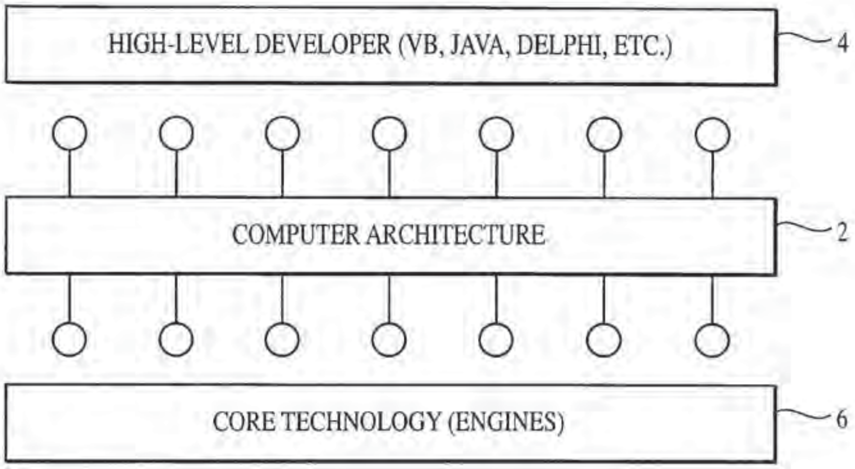

HIGH-LEVEL DEVELOPER (VB, JAVA, DELPHI, ETC.)
4
COMPUTER ARCHITECTURE
2
CORE TECHNOLOGY (ENGINES)
6

FIG. 1

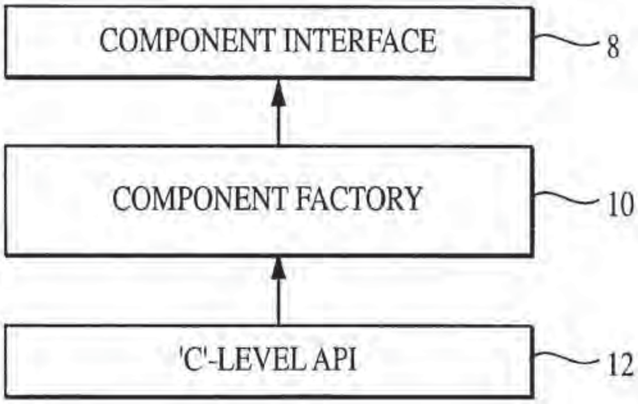

COMPONENT INTERFACE
8
COMPONENT FACTORY
10
'C'-LEVEL API
12

FIG. 2

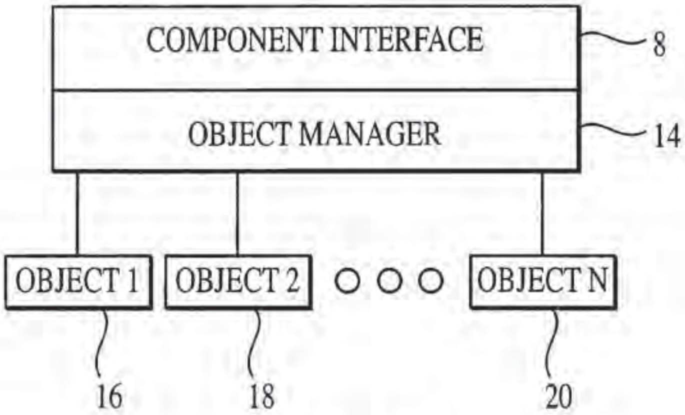

COMPONENT INTERFACE
8
OBJECT MANAGER
14
OBJECT 1
OBJECT 2
OBJECT N
16
18
20

FIG. 3

FIG. 4

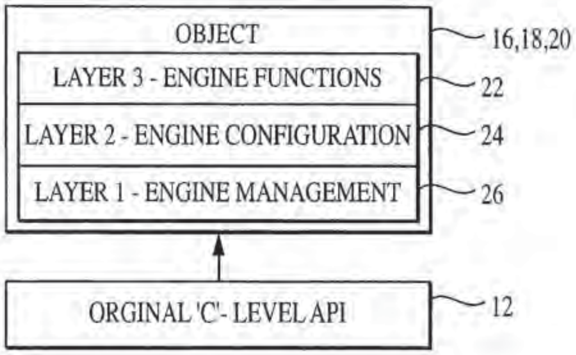

OBJECT
16,18,20
LAYER 3 - ENGINE FUNCTIONS
22
LAYER 2 - ENGINE CONFIGURATION
24
LAYER 1 - ENGINE MANAGEMENT
26
ORGINAL 'C'- LEVEL API
12

FIG. 5

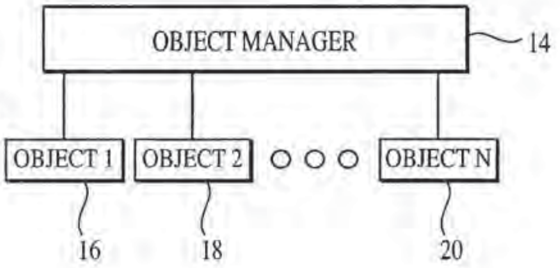
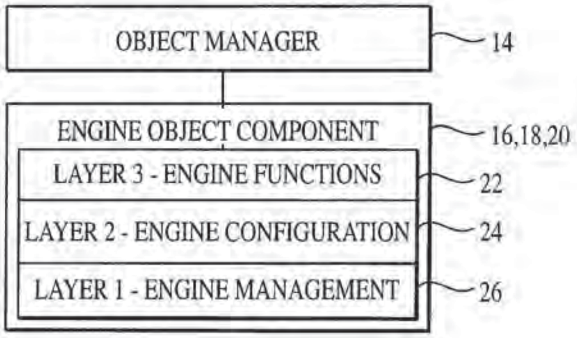
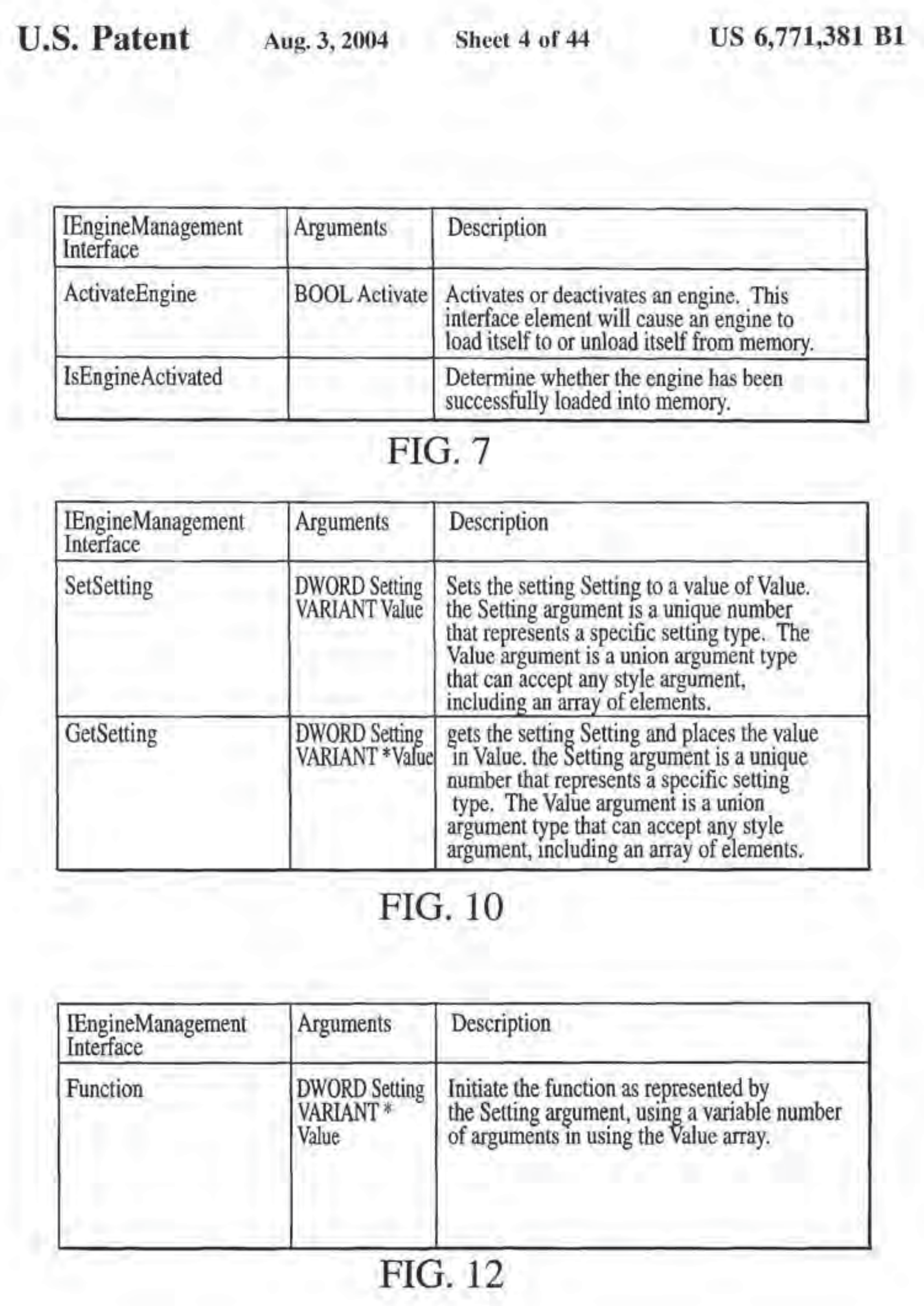


FIG. 6


OBJECT MANAGER
14
ENGINE OBJECT COMPONENT
16,18,20
LAYER 3 - ENGINE FUNCTIONS
22
LAYER 2 - ENGINE CONFIGURATION
24
LAYER 1 - ENGINE MANAGEMENT
26



IEngineManagement Interface	Arguments	Description
ActivateEngine	BOOL Activate	Activates or deactivates an engine. This interface element will cause an engine to load itself to or unload itself from memory.
IsEngineActivated		Determine whether the engine has been successfully loaded into memory.

FIG. 7

IEngineManagement Interface	Arguments	Description
SetSetting	DWORD Setting VARIANT Value	Sets the setting Setting to a value of Value. the Setting argument is a unique number that represents a specific setting type. The Value argument is a union argument type that can accept any style argument, including an array of elements.
GetSetting	DWORD Setting VARIANT *Value	gets the setting Setting and places the value in Value. the Setting argument is a unique number that represents a specific setting type. The Value argument is a union argument type that can accept any style argument, including an array of elements.

FIG. 10

IEngineManagement Interface	Arguments	Description
Function	DWORD Setting VARIANT * Value	Initiate the function as represented by the Setting argument, using a variable number of arguments in using the Value array.

FIG. 12

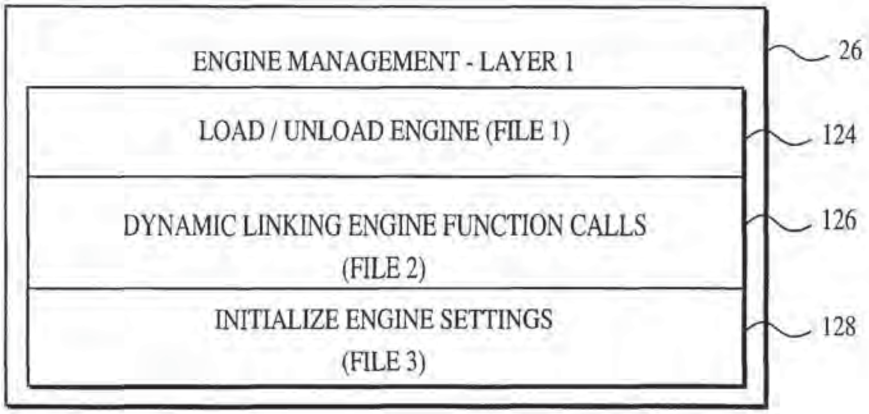
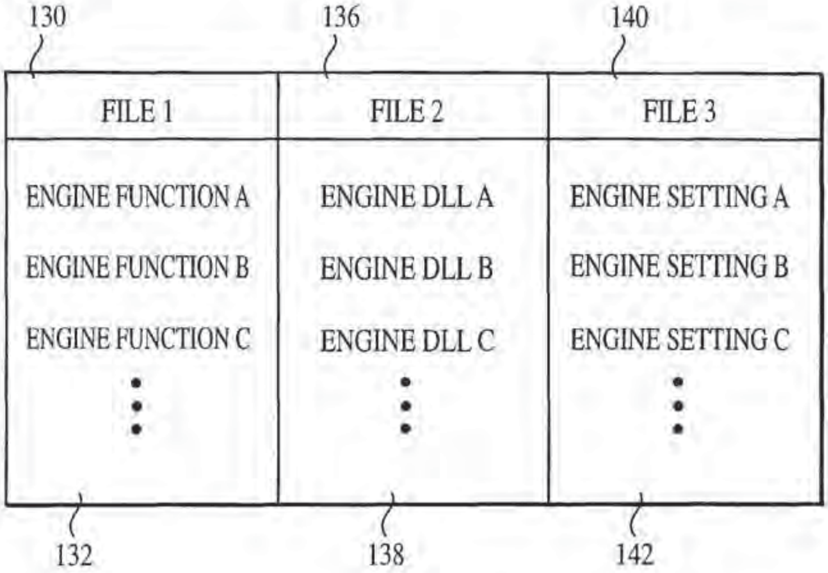

ENGINE MANAGEMENT - LAYER 1
26
LOAD / UNLOAD ENGINE (FILE 1)
124
DYNAMIC LINKING ENGINE FUNCTION CALLS (FILE 2)
126
INITIALIZE ENGINE SETTINGS (FILE 3)
128

FIG. 8



130
136
140
132
138
142

FILE 1	FILE 2	FILE 3
ENGINE FUNCTION A	ENGINE DLL A	ENGINE SETTING A
ENGINE FUNCTION B	ENGINE DLL B	ENGINE SETTING B
ENGINE FUNCTION C	ENGINE DLL C	ENGINE SETTING C
•	•	•
•	•	•
•	•	•

FIG. 9

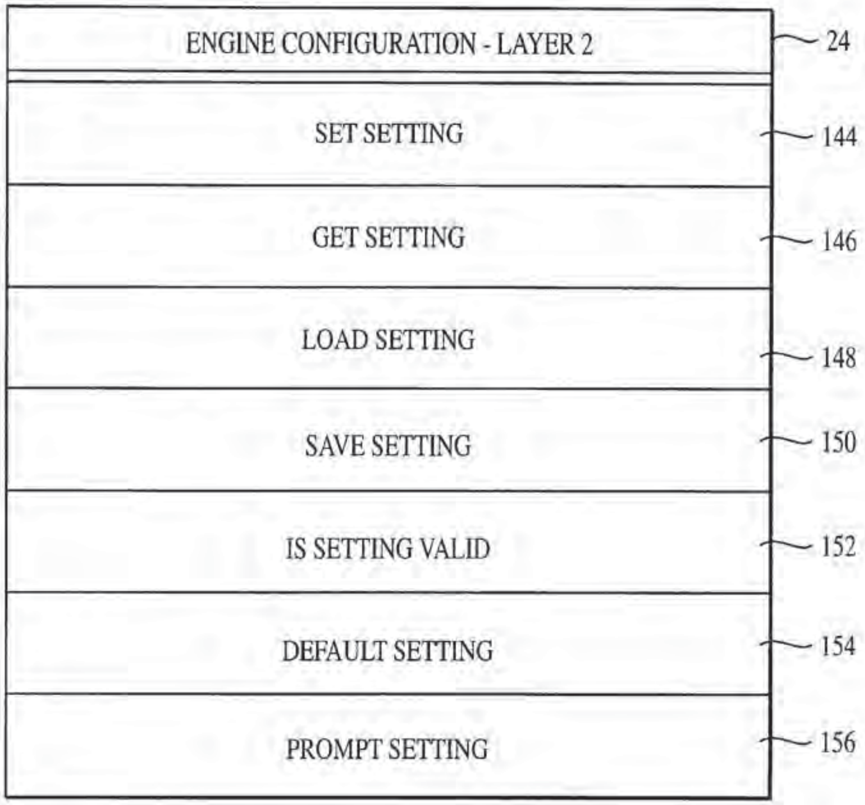

ENGINE CONFIGURATION - LAYER 2
24
SET SETTING
144
GET SETTING
146
LOAD SETTING
148
SAVE SETTING
150
IS SETTING VALID
152
DEFAULT SETTING
154
PROMPT SETTING
156

FIG. 11

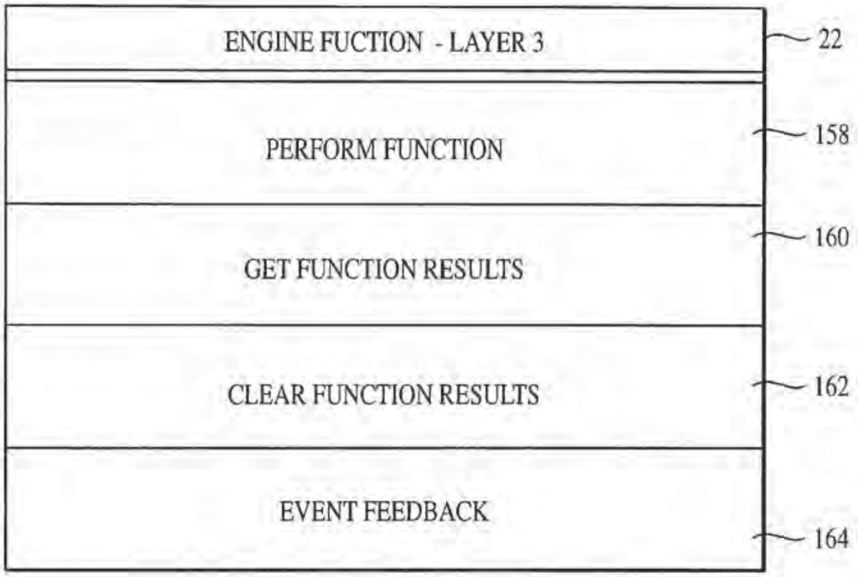

ENGINE FUCTION - LAYER 3
22
PERFORM FUNCTION
158
GET FUNCTION RESULTS
160
CLEAR FUNCTION RESULTS
162
EVENT FEEDBACK
164

FIG. 13

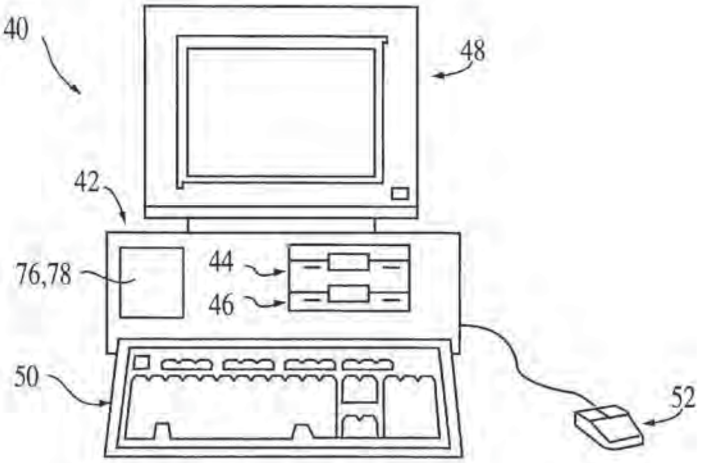

40
48
42
76,78
44
46
50
52

FIG. 14

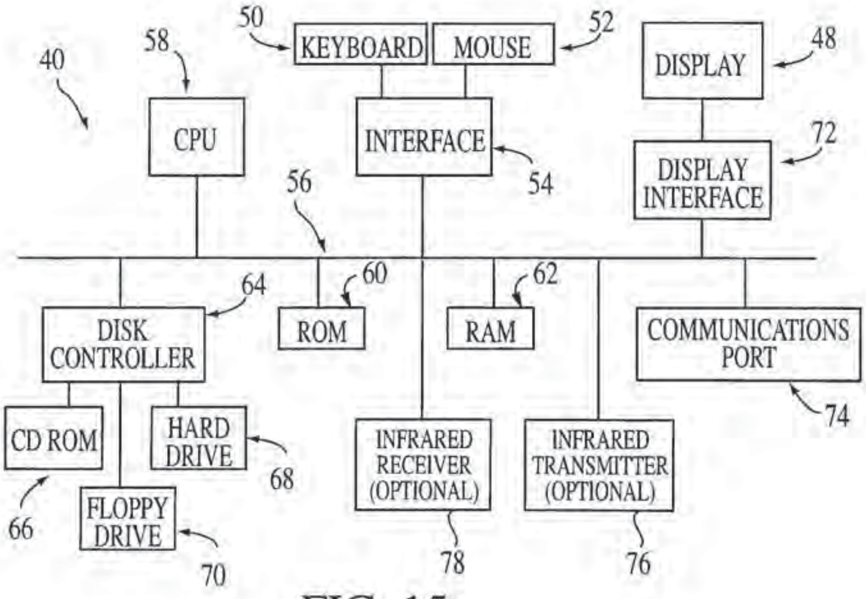

40
58
50
KEYBOARD
MOUSE
52
DISPLAY
48
CPU
INTERFACE
54
72
DISPLAY INTERFACE
56
64
60
62
DISK CONTROLLER
ROM
RAM
COMMUNICATIONS PORT
74
CD ROM
HARD DRIVE
68
INFRARED RECEIVER (OPTIONAL)
INFRARED TRANSMITTER (OPTIONAL)
66
FLOPPY DRIVE
70
78
76

FIG. 15

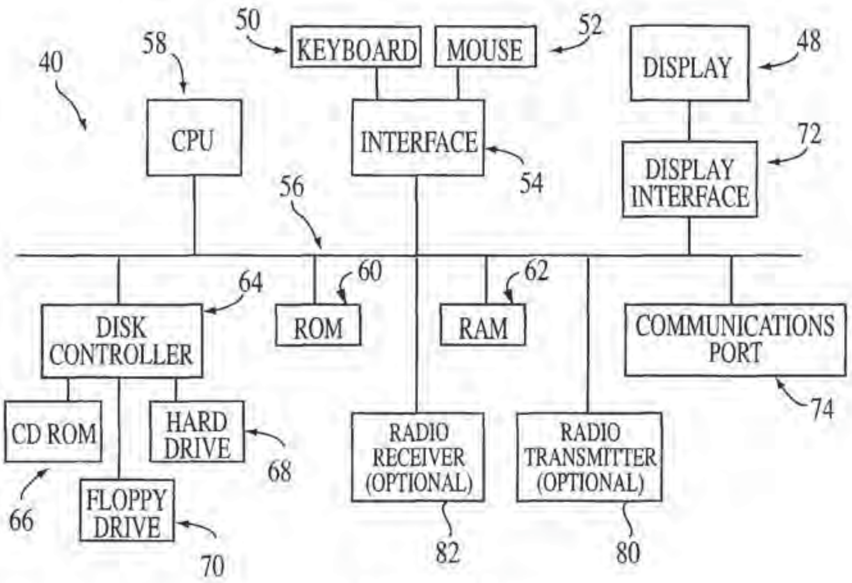

40
58
CPU
50
KEYBOARD
MOUSE
52
DISPLAY
48
INTERFACE
54
DISPLAY INTERFACE
72
56
64
DISK CONTROLLER
ROM
60
RAM
62
COMMUNICATIONS PORT
74
CD ROM
66
HARD DRIVE
68
FLOPPY DRIVE
70
RADIO RECEIVER (OPTIONAL)
82
RADIO TRANSMITTER (OPTIONAL)
80

FIG. 16

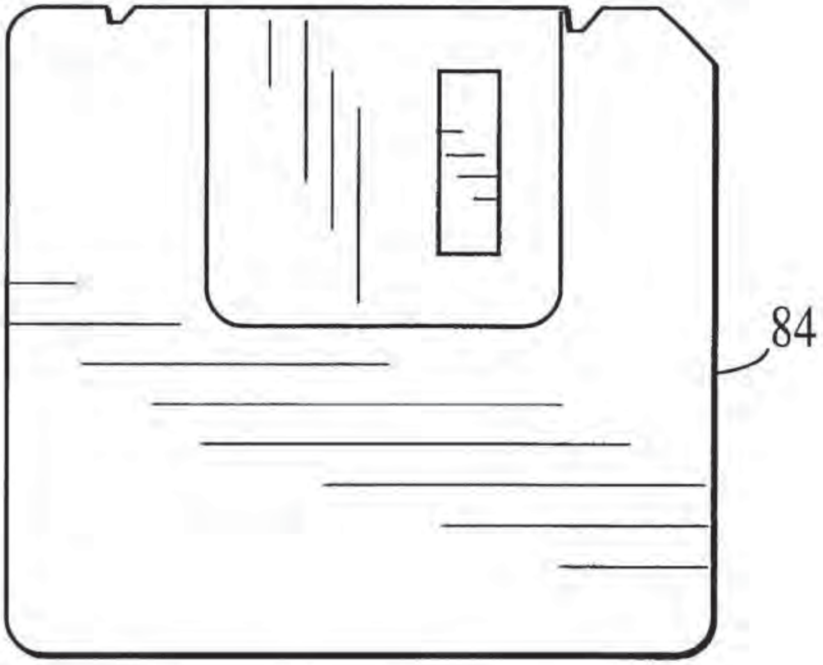


FIG. 17

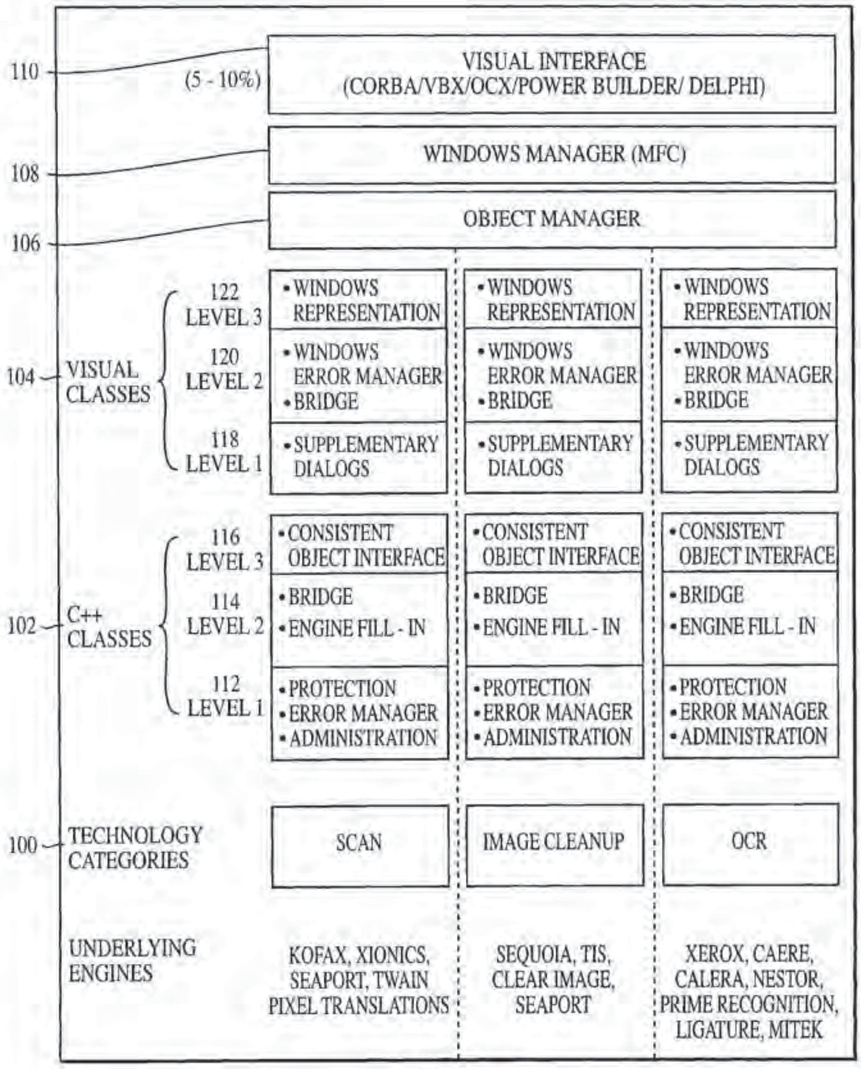

110
(5 - 10%)
VISUAL INTERFACE (CORBA/VBX/OCX/POWER BUILDER/ DELPHI)
108
WINDOWS MANAGER (MFC)
106
OBJECT MANAGER
104
VISUAL CLASSES
122 LEVEL 3
•WINDOWS REPRESENTATION
120 LEVEL 2
•WINDOWS ERROR MANAGER
•BRIDGE
118 LEVEL 1
•SUPPLEMENTARY DIALOGS
102
C++ CLASSES
116 LEVEL 3
•CONSISTENT OBJECT INTERFACE
114 LEVEL 2
•BRIDGE
•ENGINE FILL - IN
112 LEVEL 1
•PROTECTION
•ERROR MANAGER
•ADMINISTRATION
100
TECHNOLOGY CATEGORIES
SCAN
IMAGE CLEANUP
OCR
UNDERLYING ENGINES
KOFAX, XIONICS, SEAPORT, TWAIN PIXEL TRANSLATIONS
SEQUOIA, TIS, CLEAR IMAGE, SEAPORT
XEROX, CAERE, CALERA, NESTOR, PRIME RECOGNITION, LIGATURE, MITEK

FIG. 18

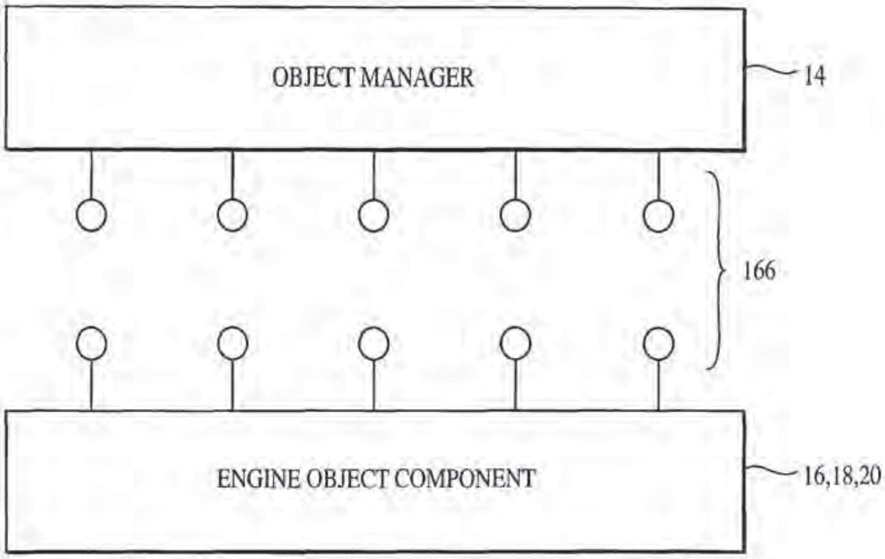


FIG. 19

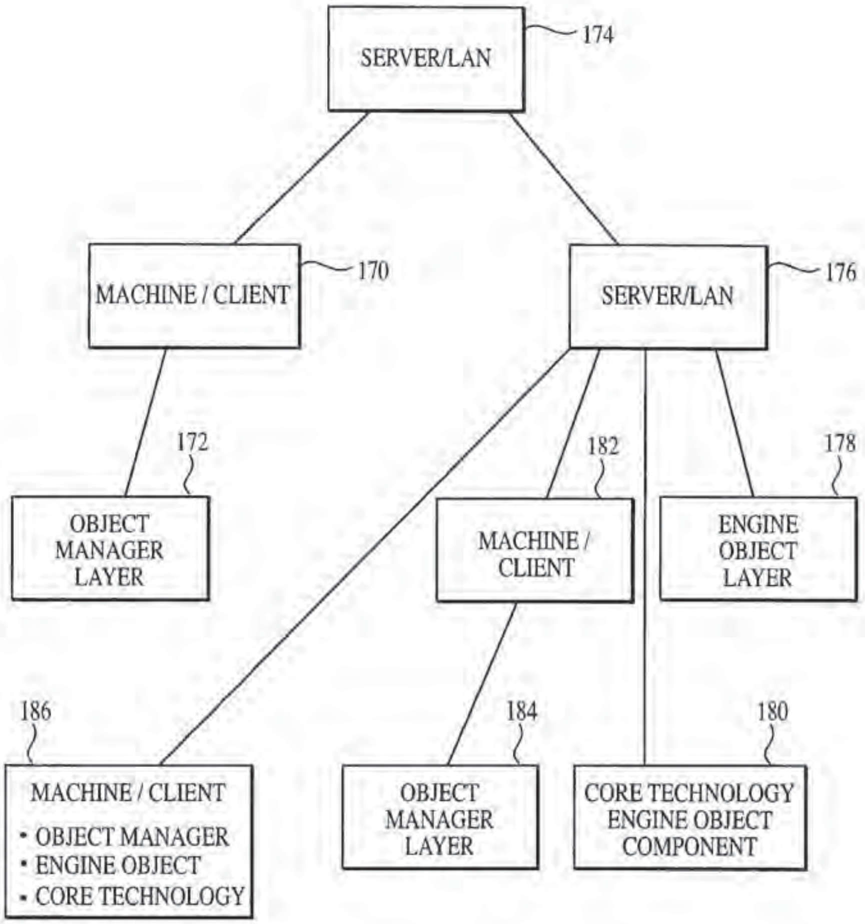

SERVER/LAN
174
MACHINE / CLIENT
170
SERVER/LAN
176
172
OBJECT MANAGER LAYER
182
MACHINE / CLIENT
178
ENGINE OBJECT LAYER
186
184
180
MACHINE / CLIENT
• OBJECT MANAGER
• ENGINE OBJECT
• CORE TECHNOLOGY
OBJECT MANAGER LAYER
CORE TECHNOLOGY ENGINE OBJECT COMPONENT

FIG. 20

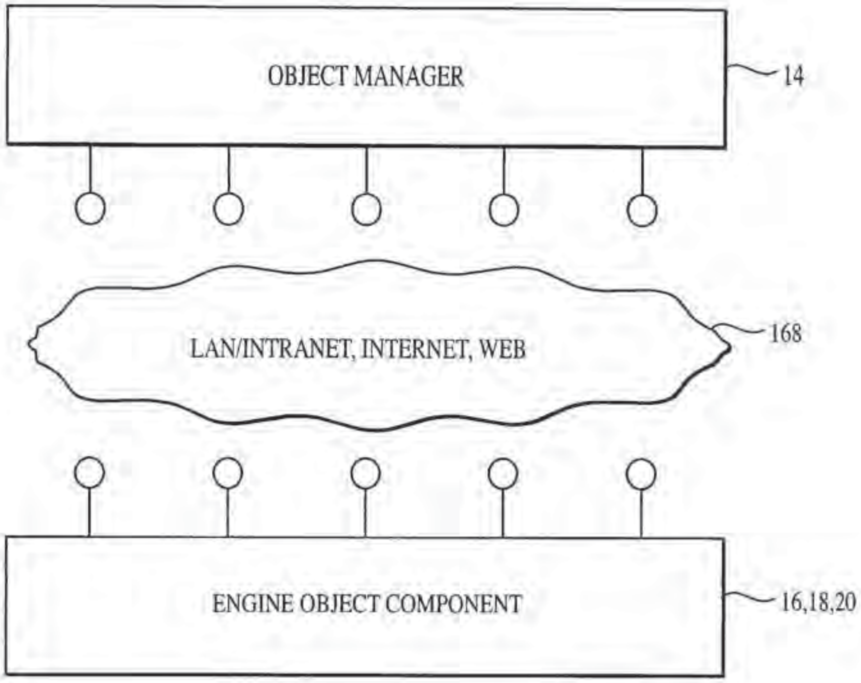

OBJECT MANAGER
14
LAN/INTRANET, INTERNET, WEB
168
ENGINE OBJECT COMPONENT
16,18,20

FIG. 21

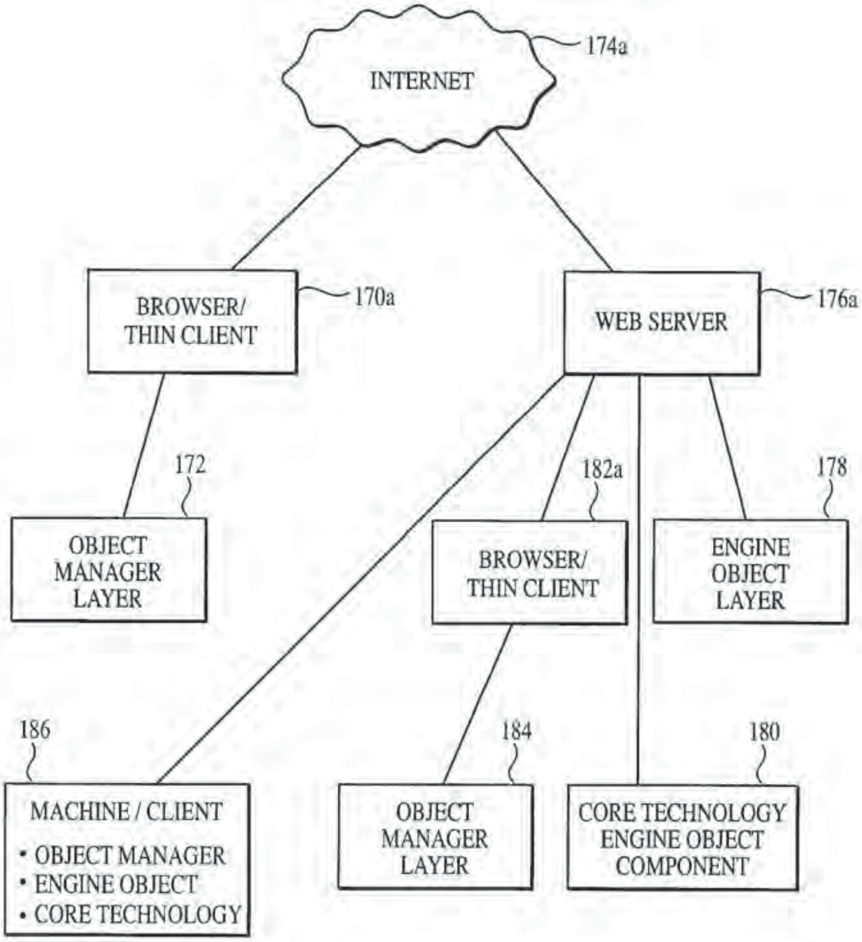

INTERNET
174a
BROWSER/ THIN CLIENT
170a
WEB SERVER
176a
172
OBJECT MANAGER LAYER
182a
BROWSER/ THIN CLIENT
178
ENGINE OBJECT LAYER
186
MACHINE / CLIENT
• OBJECT MANAGER
• ENGINE OBJECT
• CORE TECHNOLOGY
184
OBJECT MANAGER LAYER
180
CORE TECHNOLOGY ENGINE OBJECT COMPONENT

FIG. 22

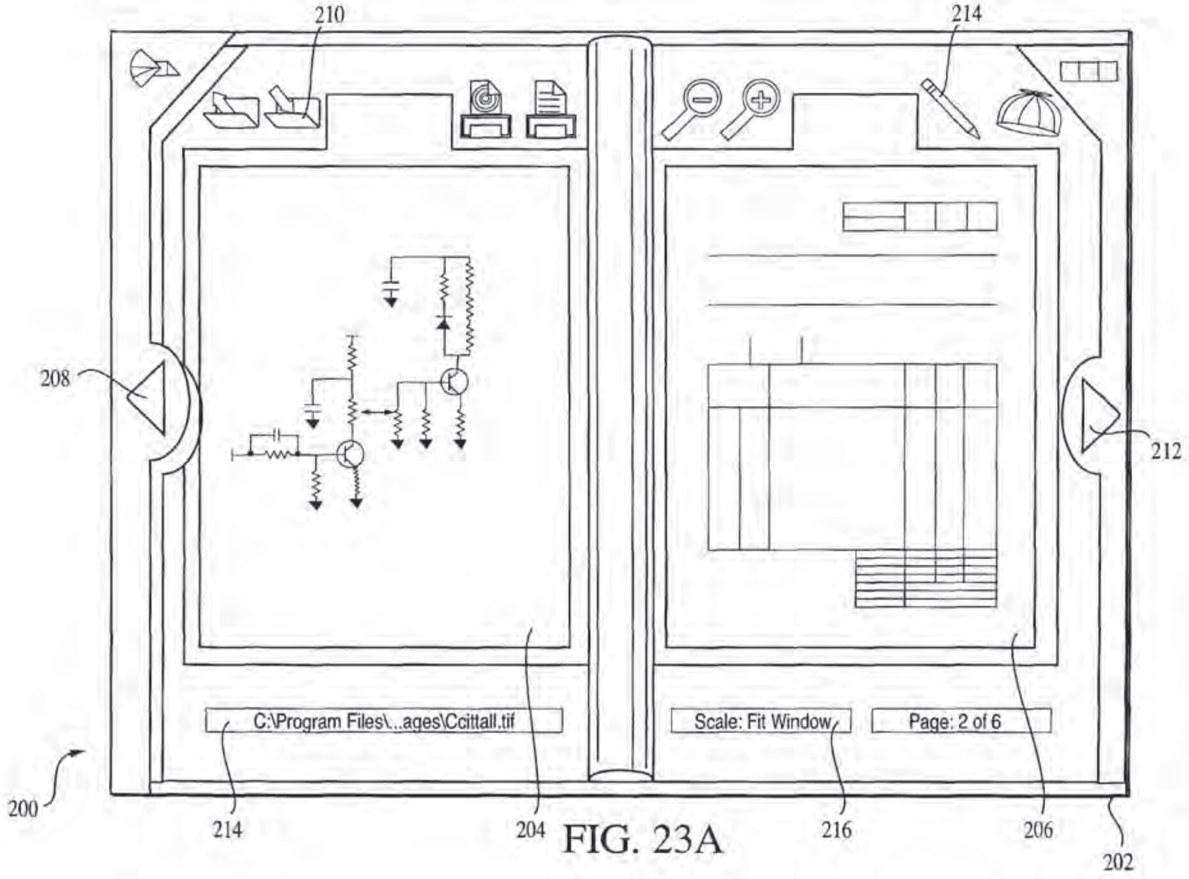
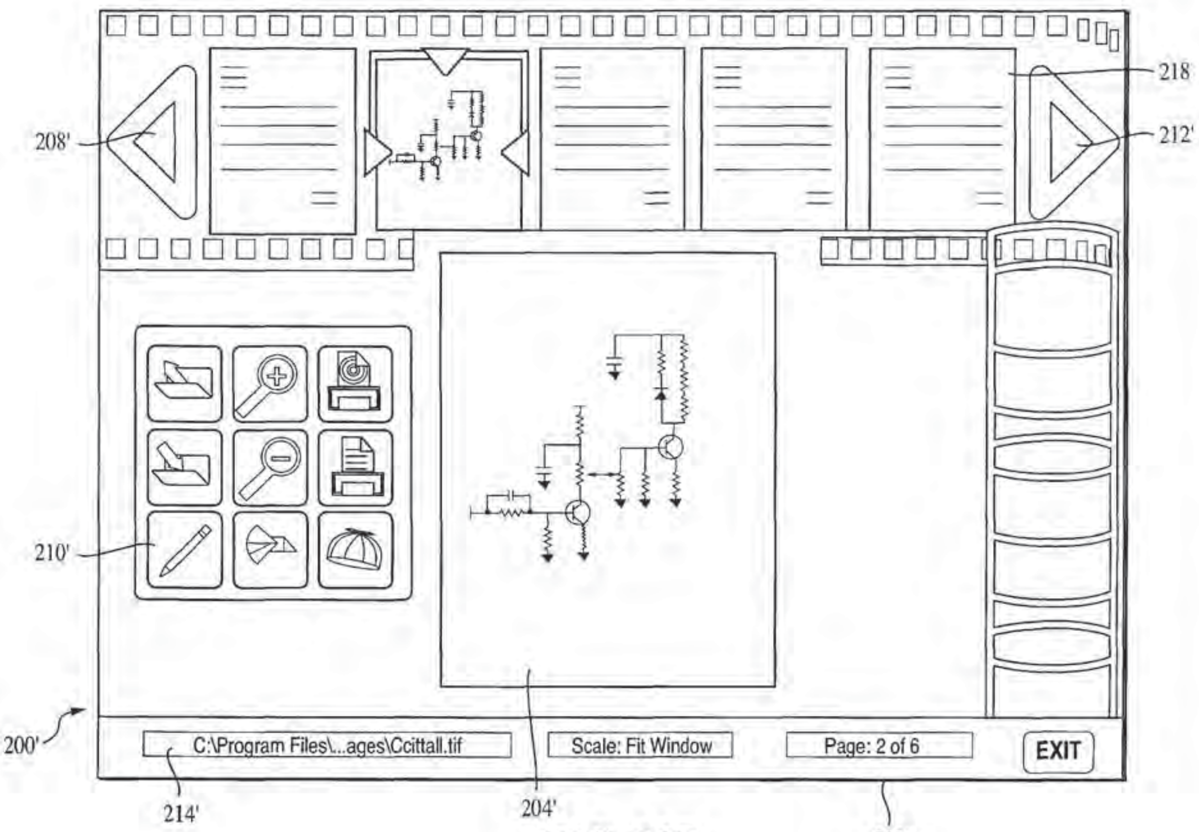

210
214
208
212
C:\Program Files\...ages\Ccittall.tif
Scale: Fit Window
Page: 2 of 6
200
214
204
216
206
202

FIG. 23A



C:\Program Files\...ages\Ccittall.tif

Scale: Fit Window

Page: 2 of 6

FIG. 23B

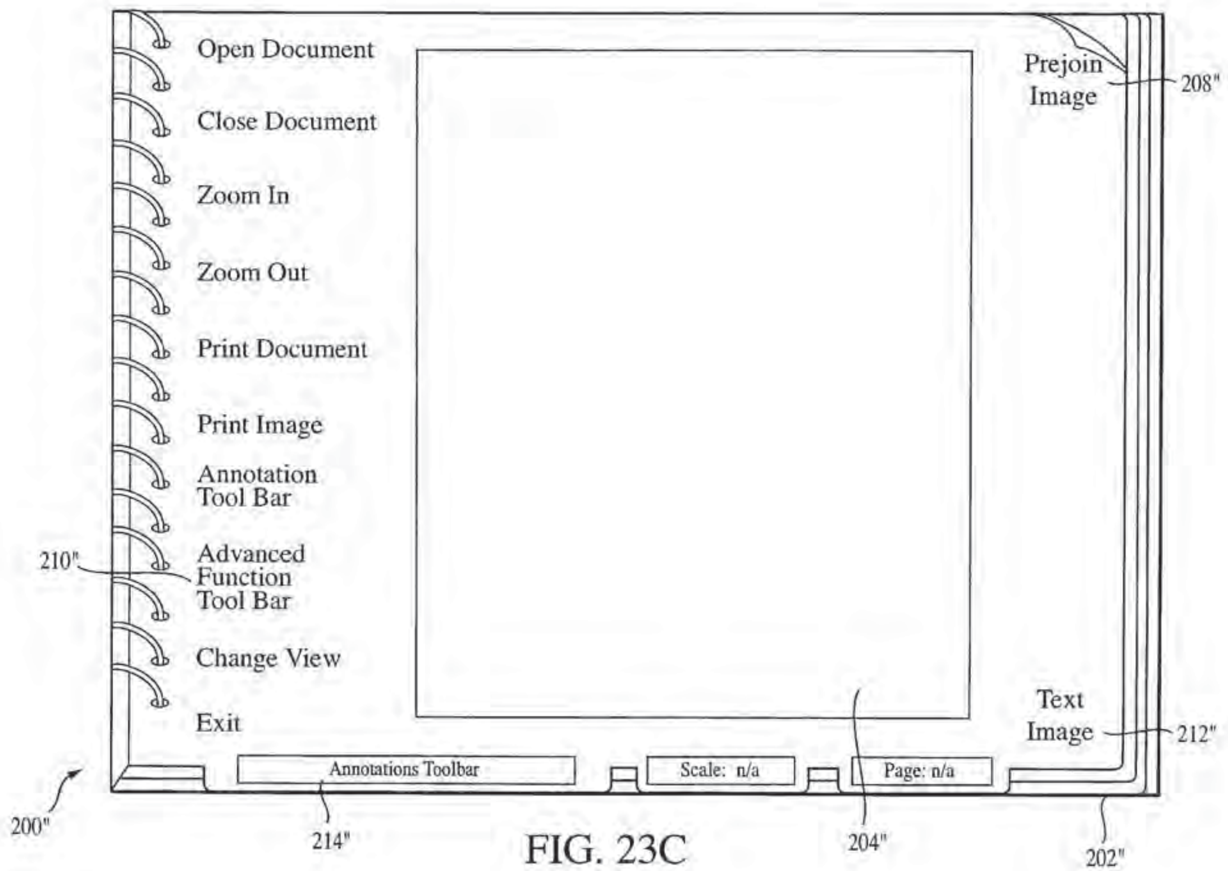

Open Document
Close Document
Zoom In
Zoom Out
Print Document
Print Image
Annotation Tool Bar
Advanced Function Tool Bar
Change View
Exit
Prejoin Image
208"
Text Image
212"
210"
Annotations Toolbar
Scale: n/a
Page: n/a
200"
214"
204"
202"

FIG. 23C

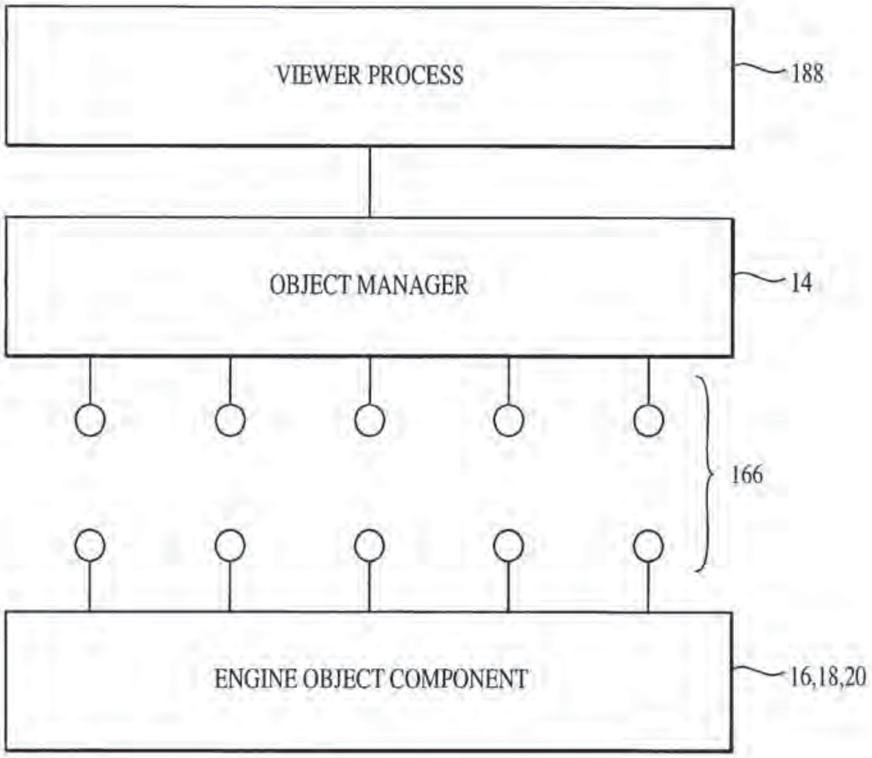

VIEWER PROCESS
188
OBJECT MANAGER
14
166
ENGINE OBJECT COMPONENT
16,18,20

FIG. 24

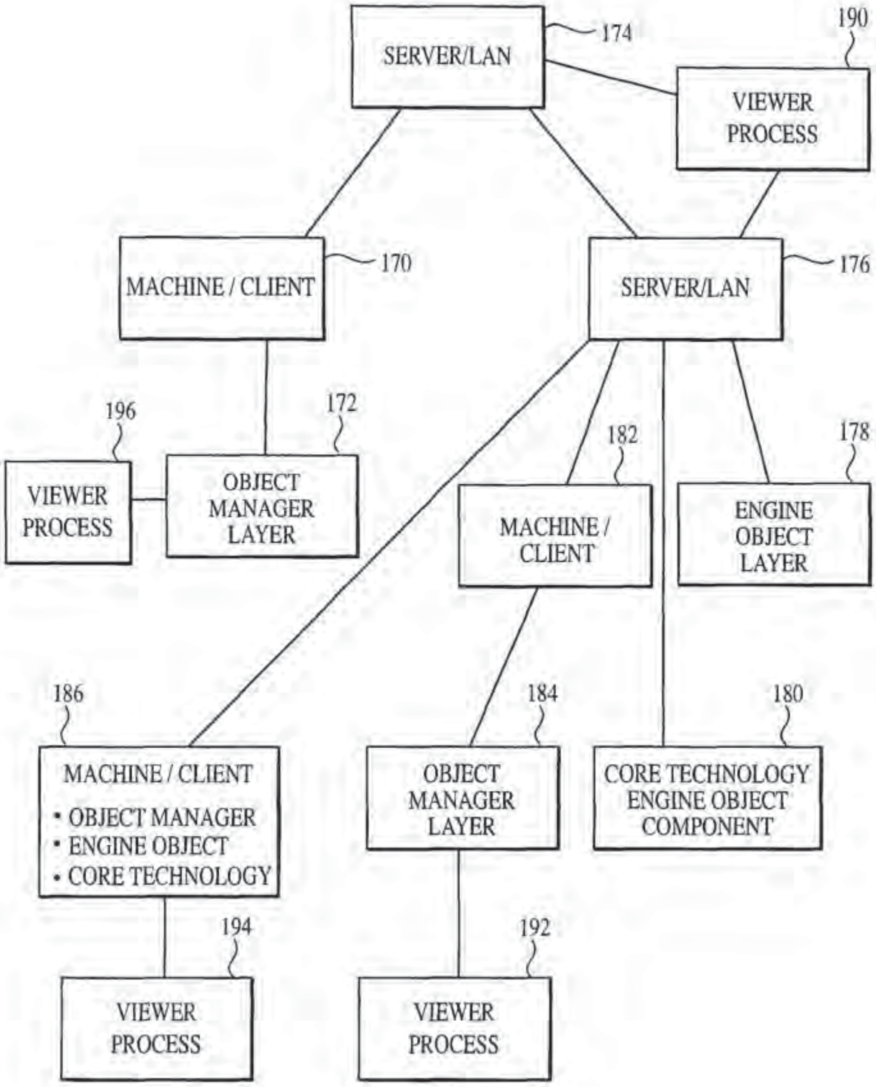

SERVER/LAN
174
190
VIEWER PROCESS
MACHINE / CLIENT
170
SERVER/LAN
176
196
172
182
178
VIEWER PROCESS
OBJECT MANAGER LAYER
MACHINE / CLIENT
ENGINE OBJECT LAYER
186
184
180
MACHINE / CLIENT
• OBJECT MANAGER
• ENGINE OBJECT
• CORE TECHNOLOGY
OBJECT MANAGER LAYER
CORE TECHNOLOGY ENGINE OBJECT COMPONENT
194
192
VIEWER PROCESS
VIEWER PROCESS

FIG. 25

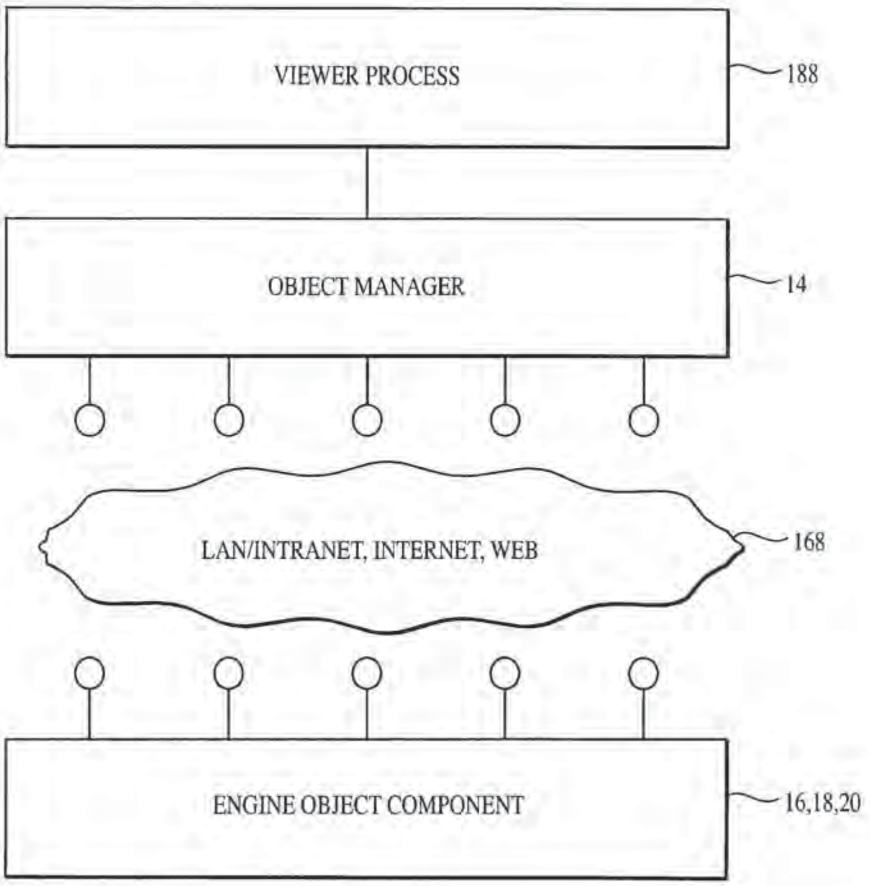

VIEWER PROCESS
188
OBJECT MANAGER
14
LAN/INTRANET, INTERNET, WEB
168
ENGINE OBJECT COMPONENT
16,18,20

FIG. 26

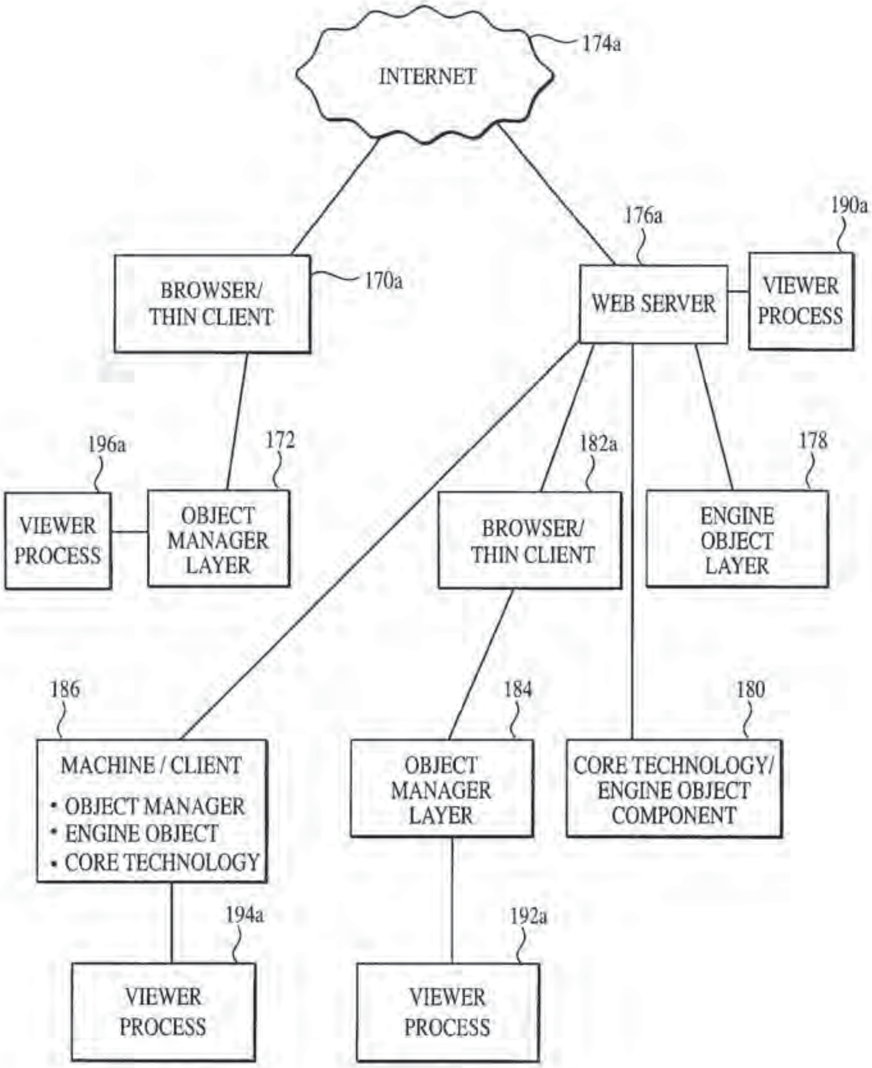

INTERNET
174a
176a
190a
BROWSER/ THIN CLIENT
170a
WEB SERVER
VIEWER PROCESS
196a
172
182a
178
VIEWER PROCESS
OBJECT MANAGER LAYER
BROWSER/ THIN CLIENT
ENGINE OBJECT LAYER
186
184
180
MACHINE / CLIENT
• OBJECT MANAGER
• ENGINE OBJECT
• CORE TECHNOLOGY
OBJECT MANAGER LAYER
CORE TECHNOLOGY/ ENGINE OBJECT COMPONENT
194a
192a
VIEWER PROCESS
VIEWER PROCESS

FIG. 27

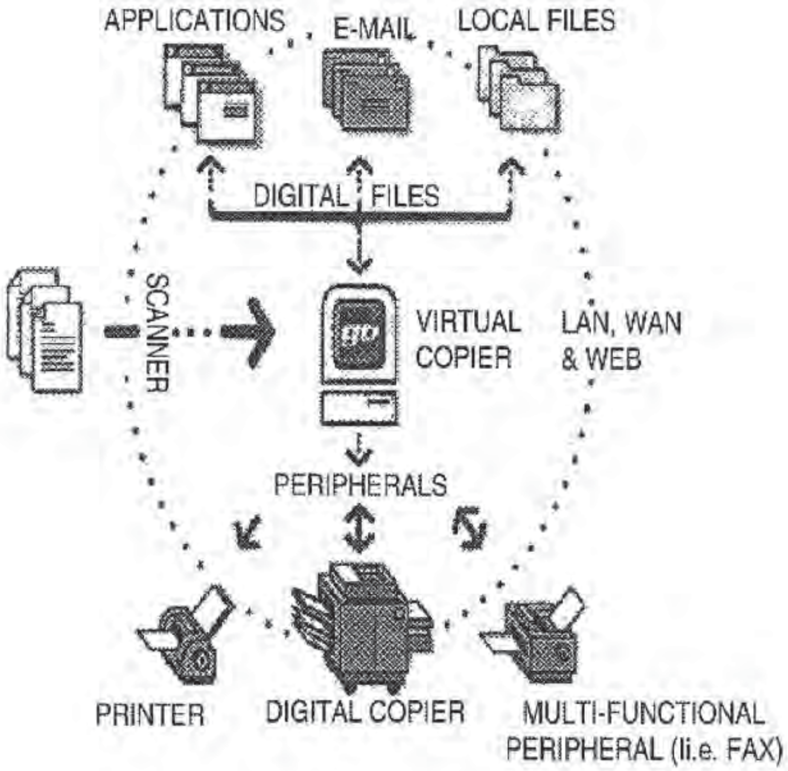
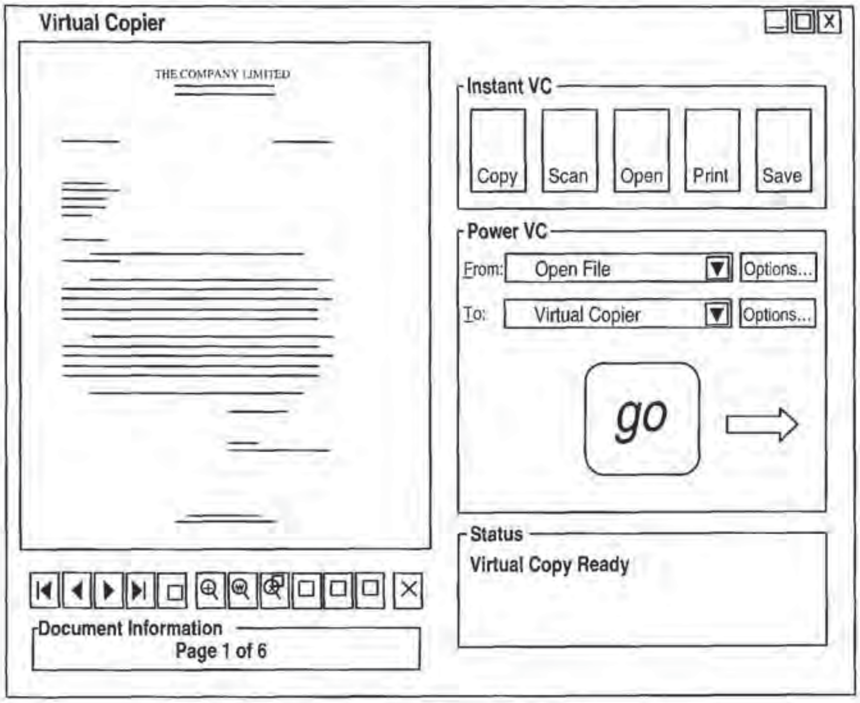

APPLICATIONS
E-MAIL
LOCAL FILES
DIGITAL FILES
SCANNER
VIRTUAL COPIER
LAN, WAN & WEB
PERIPHERALS
PRINTER
DIGITAL COPIER
MULTI-FUNCTIONAL PERIPHERAL (Ii.e. FAX)

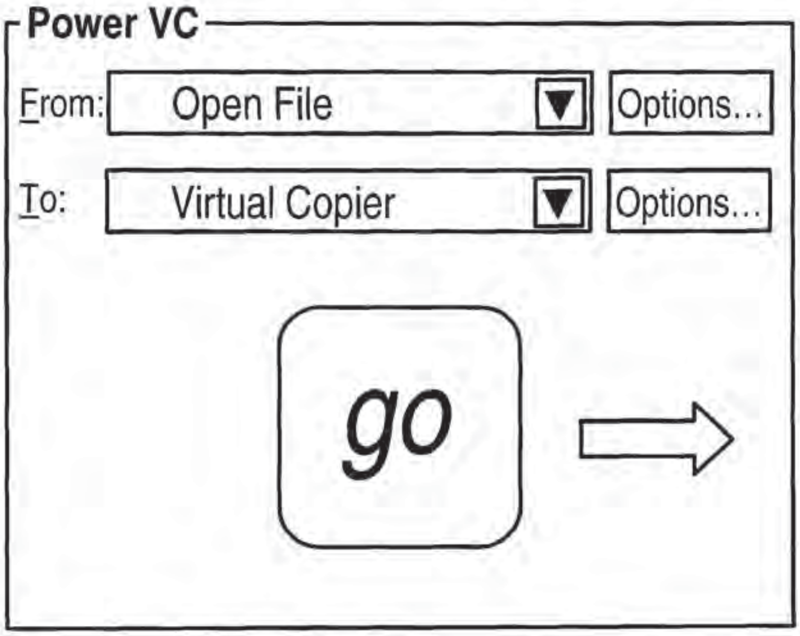
FIG. 28



Virtual Copy Ready

Page 1 of 6

FIG. 29



Power VC

From: Open File

To: Virtual Copier

FIG. 30

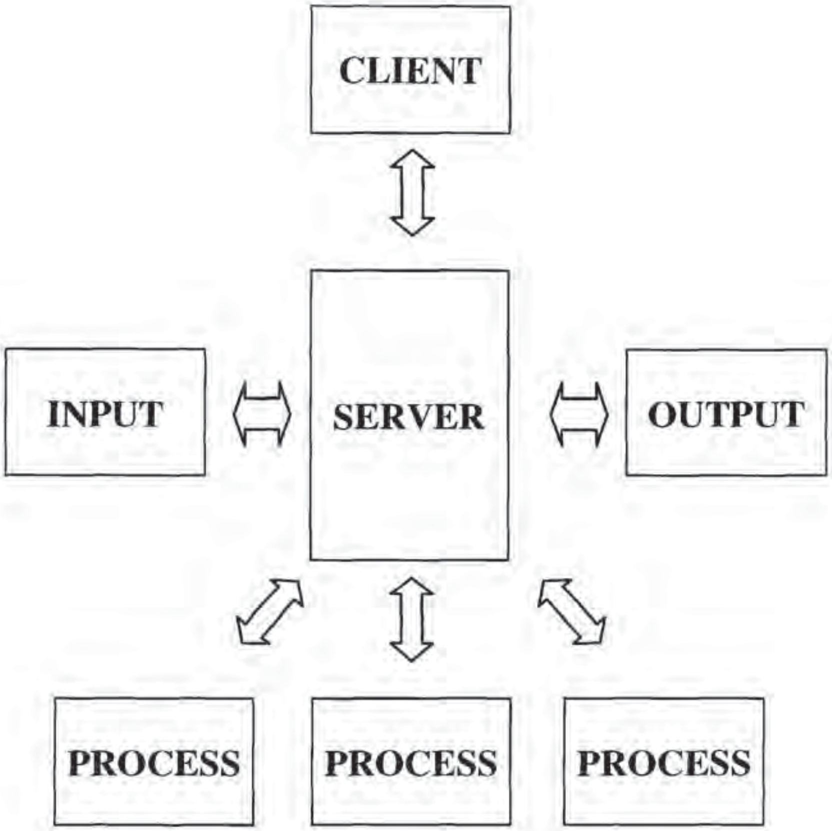


FIG. 31

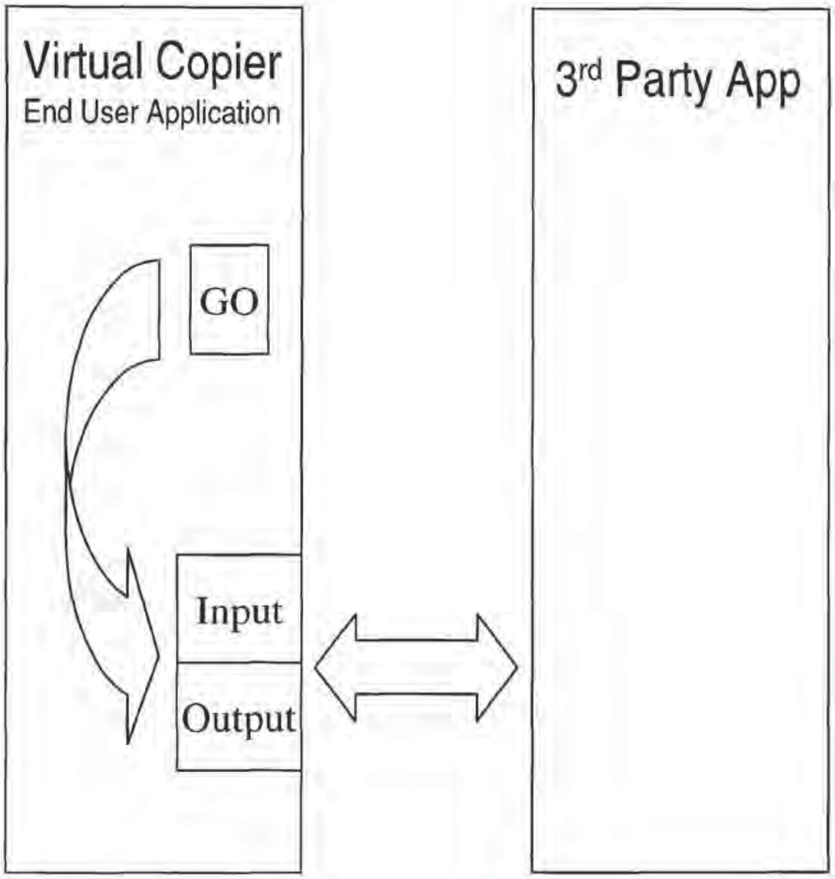

Virtual Copier
End User Application
GO
Input
Output
3rd Party App

FIG. 32

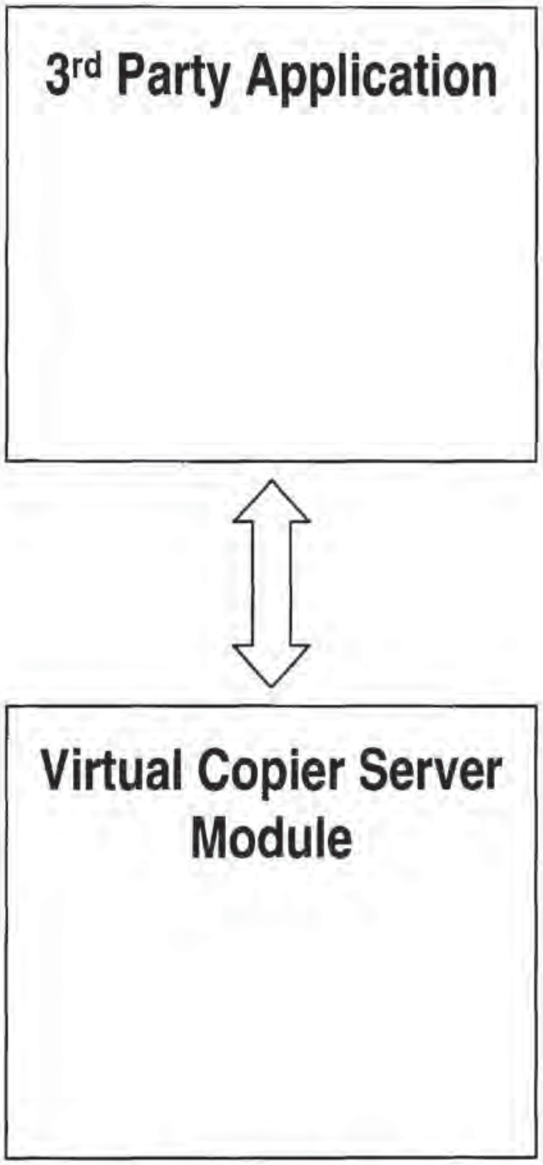


FIG. 33

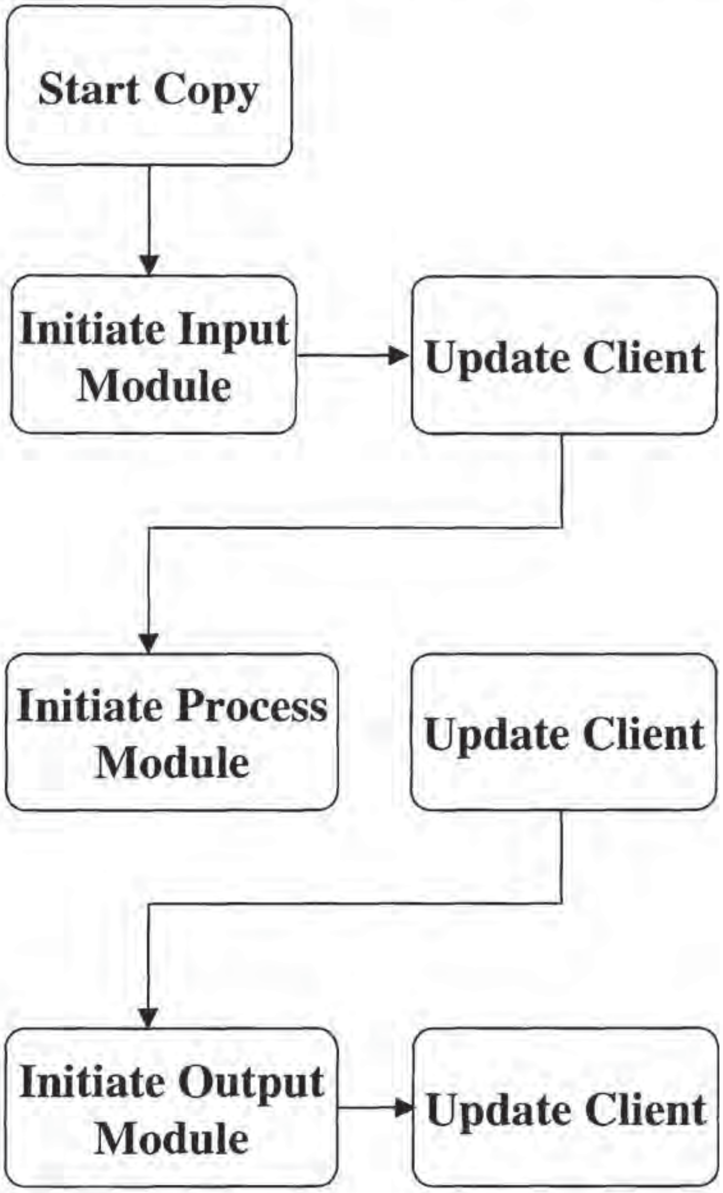

Start Copy
Initiate Input Module
Update Client
Initiate Process Module
Update Client
Initiate Output Module
Update Client

FIG. 34

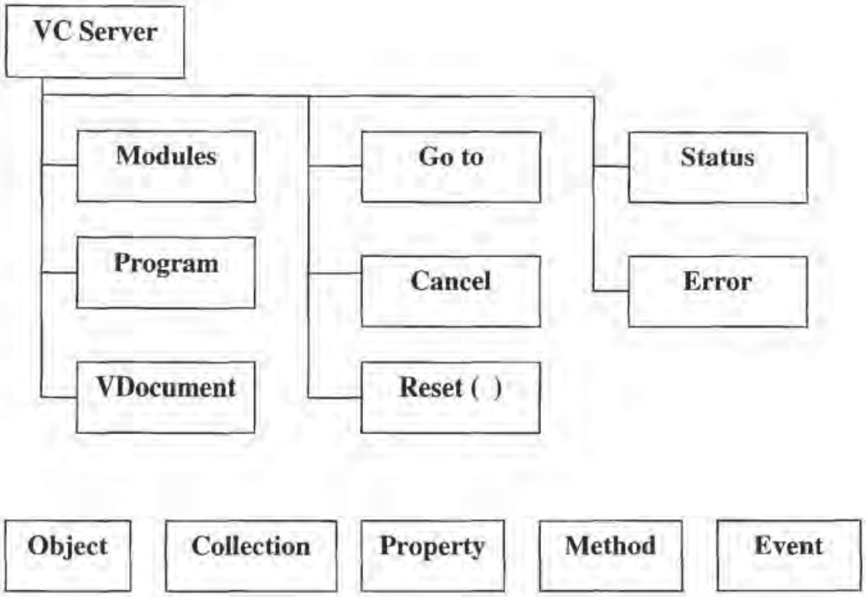

VC Server
Modules
Program
VDocument
Go to
Cancel
Reset ( )
Status
Error
Object
Collection
Property
Method
Event

FIG. 35

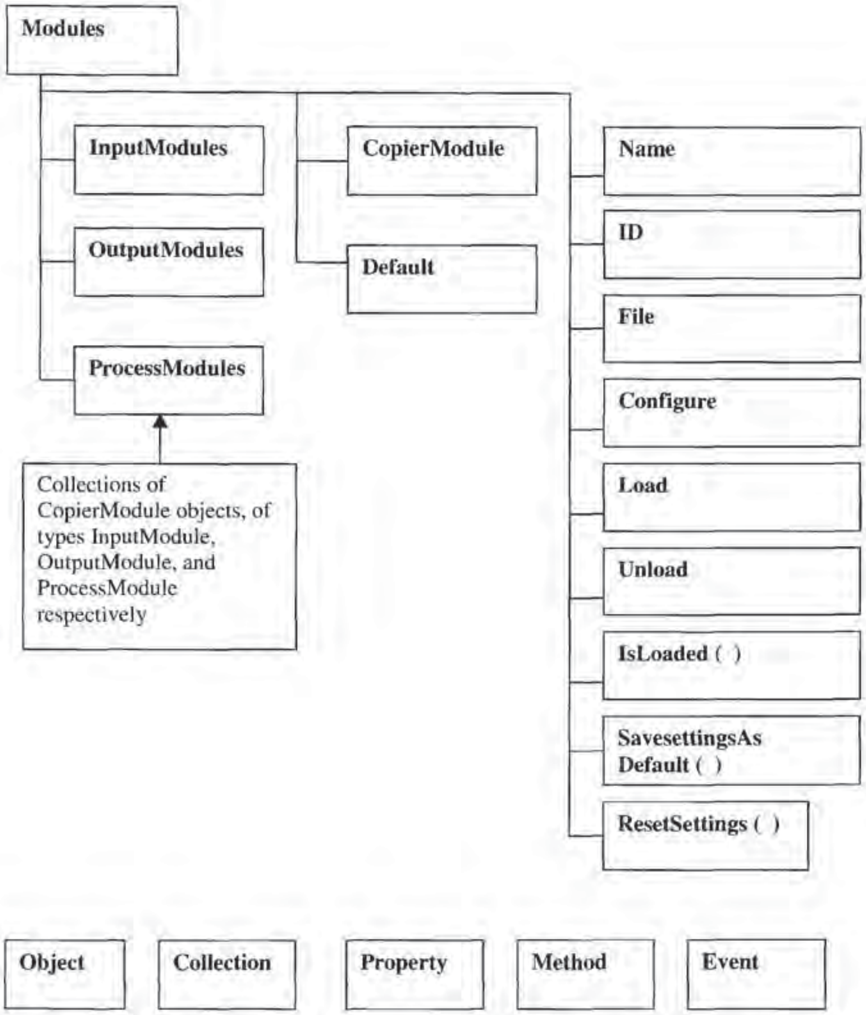

Modules
InputModules
OutputModules
ProcessModules
Collections of CopierModule objects, of types InputModule, OutputModule, and ProcessModule respectively
CopierModule
Default
Name
ID
File
Configure
Load
Unload
IsLoaded ( )
SavesettingsAs Default ( )
ResetSettings ( )
Object
Collection
Property
Method
Event

FIG. 36

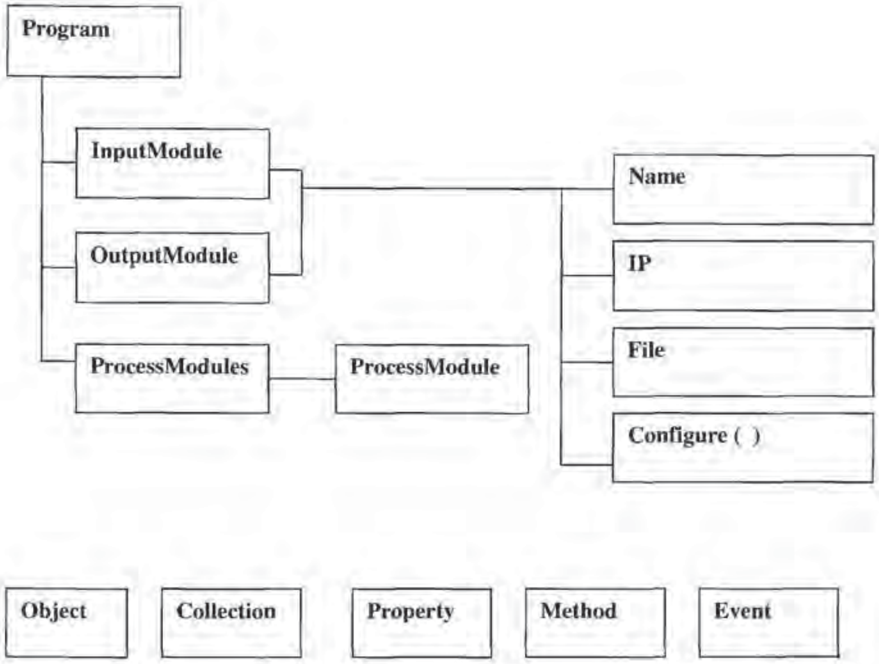
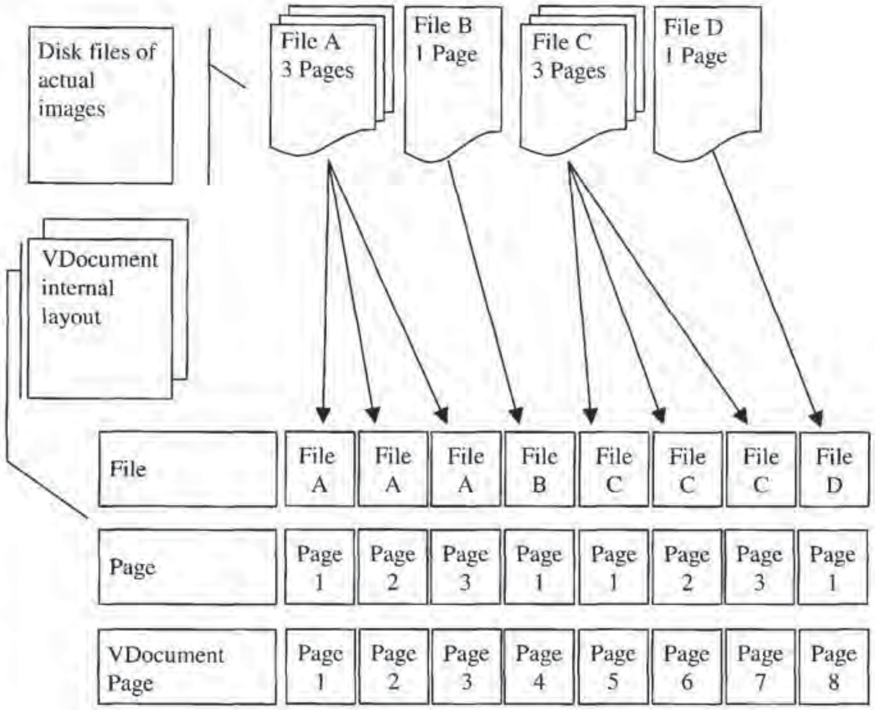

Program
InputModule
OutputModule
ProcessModules
ProcessModule
Name
IP
File
Configure ( )
Object
Collection
Property
Method
Event

FIG. 37



File	File A	File A	File A	File B	File C	File C	File C	File D
Page	Page 1	Page 2	Page 3	Page 1	Page 1	Page 2	Page 3	Page 1
VDocument Page	Page 1	Page 2	Page 3	Page 4	Page 5	Page 6	Page 7	Page 8

FIG. 38

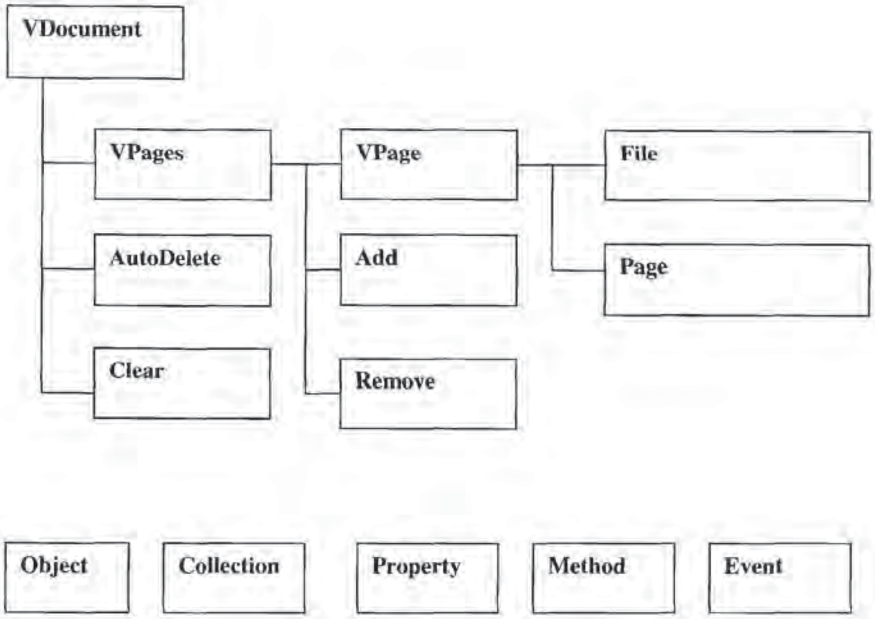

VDocument
VPages
AutoDelete
Clear
VPage
Add
Remove
File
Page
Object
Collection
Property
Method
Event

FIG. 39

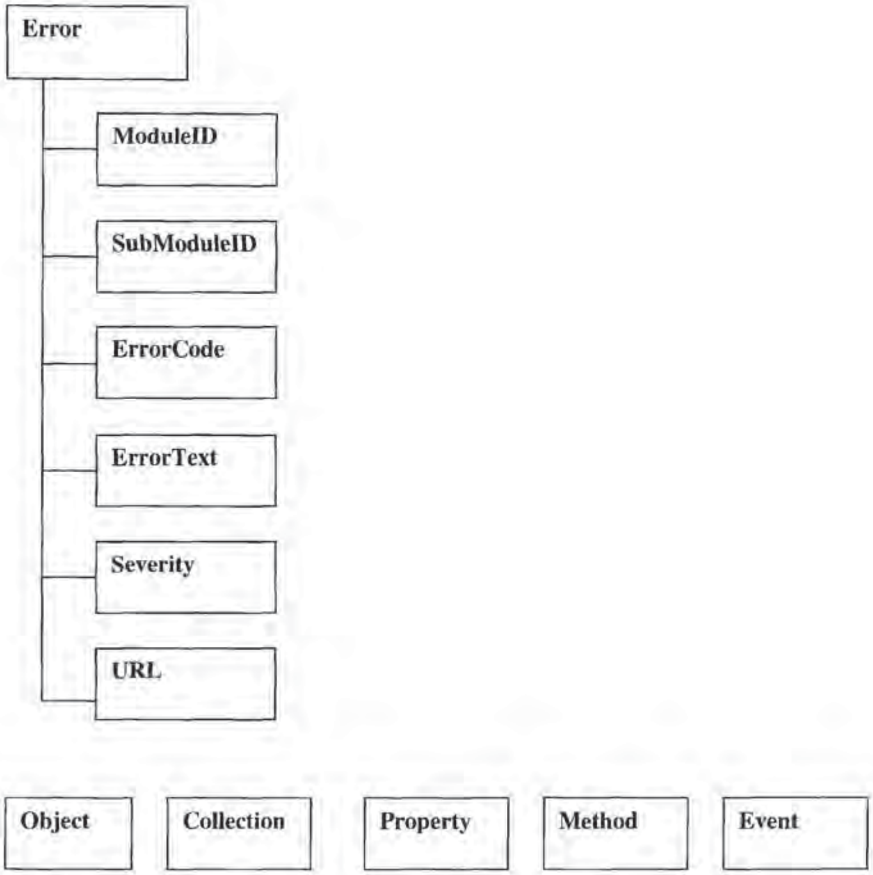

Error
ModuleID
SubModuleID
ErrorCode
ErrorText
Severity
URL
Object
Collection
Property
Method
Event

FIG. 40

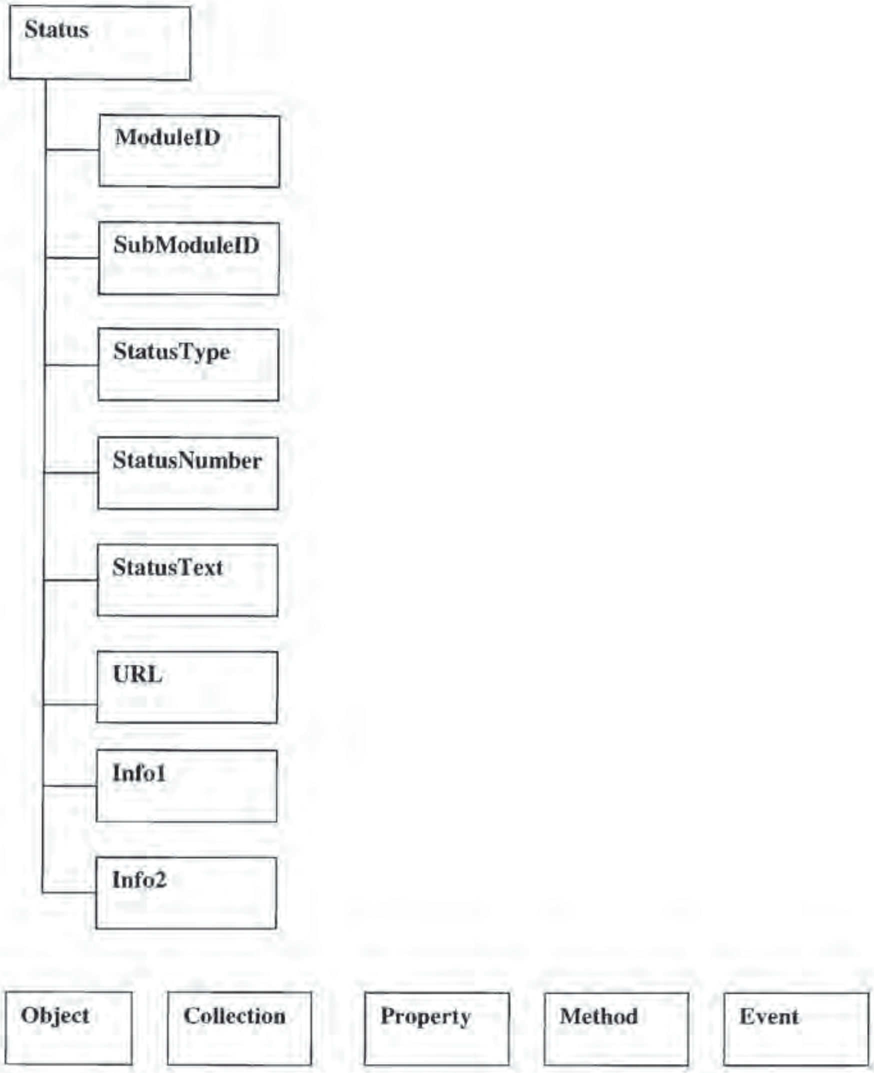

Status
ModuleID
SubModuleID
StatusType
StatusNumber
StatusText
URL
Info1
Info2
Object
Collection
Property
Method
Event

FIG. 41

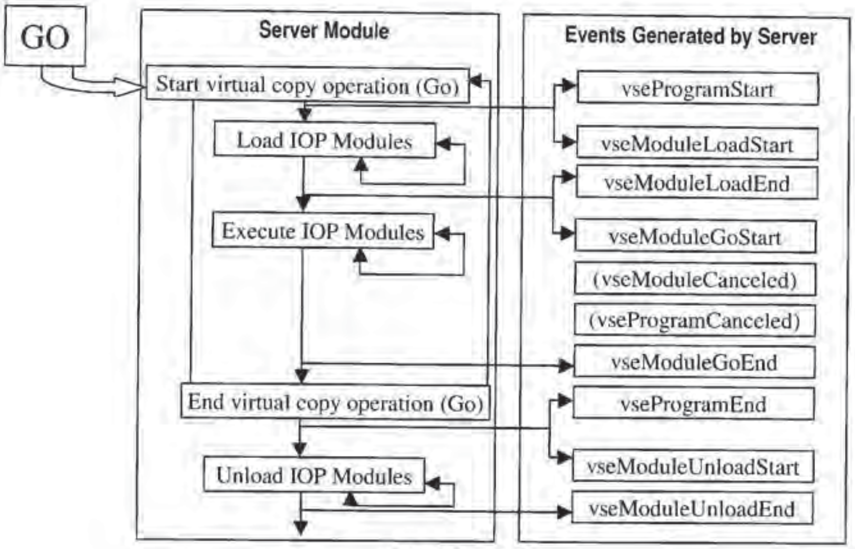

GO
Server Module
Start virtual copy operation (Go)
Load IOP Modules
Execute IOP Modules
End virtual copy operation (Go)
Unload IOP Modules
Events Generated by Server
vseProgramStart
vseModuleLoadStart
vseModuleLoadEnd
vseModuleGoStart
(vseModuleCanceled)
(vseProgramCanceled)
vseModuleGoEnd
vseProgramEnd
vseModuleUnloadStart
vseModuleUnloadEnd

FIG. 42

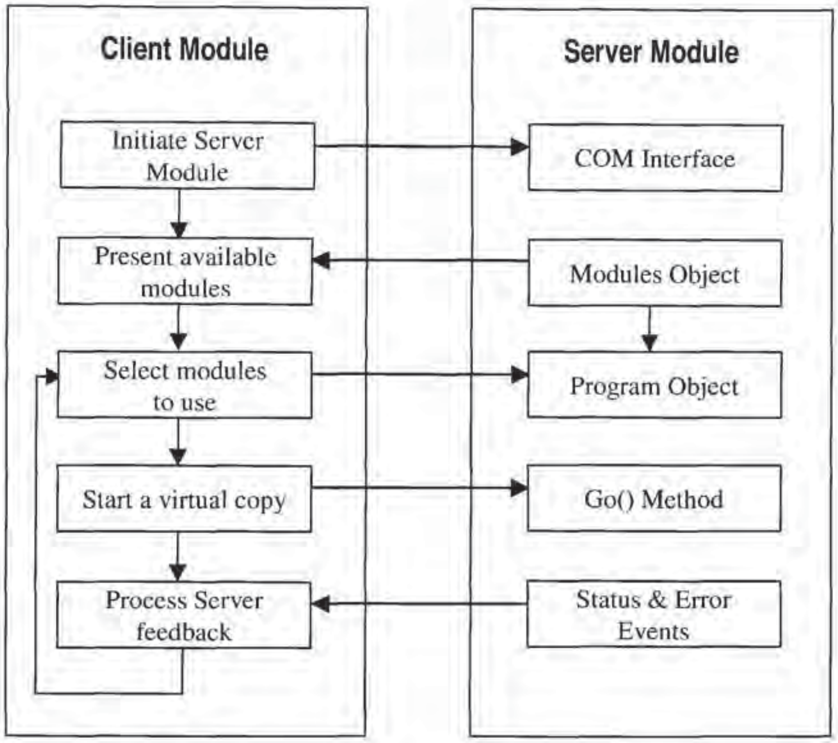
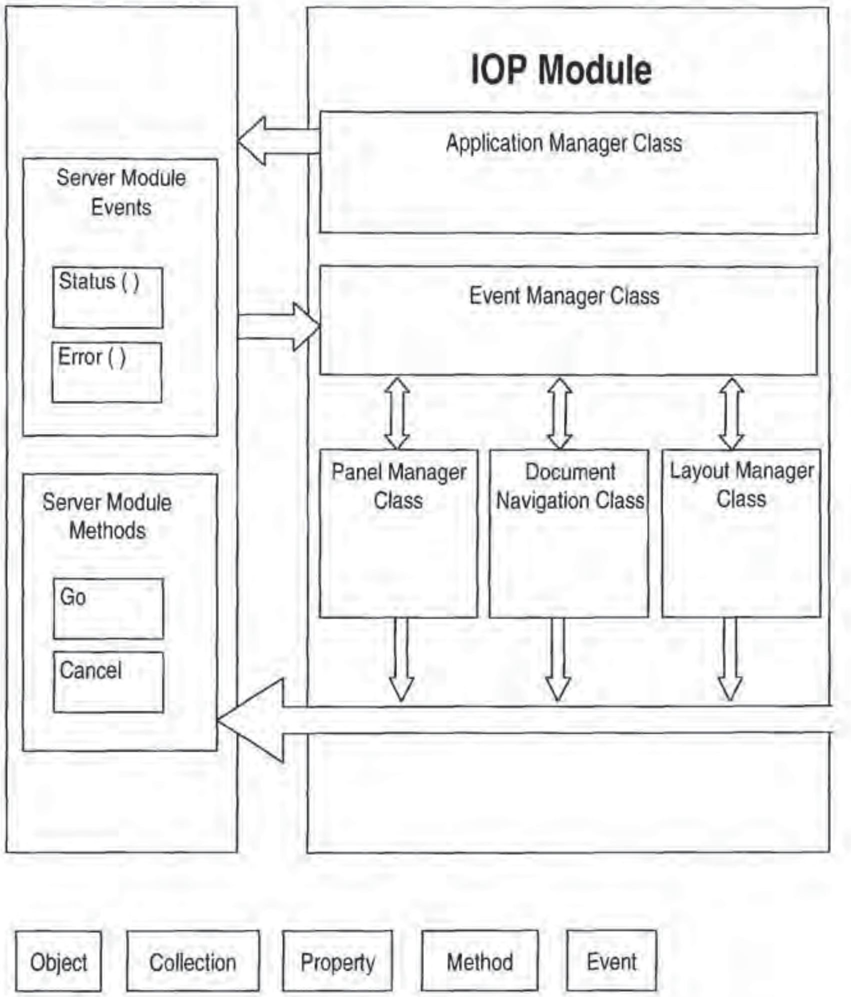

Client Module
Server Module
Initiate Server Module
COM Interface
Present available modules
Modules Object
Select modules to use
Program Object
Start a virtual copy
Go() Method
Process Server feedback
Status & Error Events

FIG. 43



The Client Module has a fixed set of features that it needs to perform:

FIG. 44

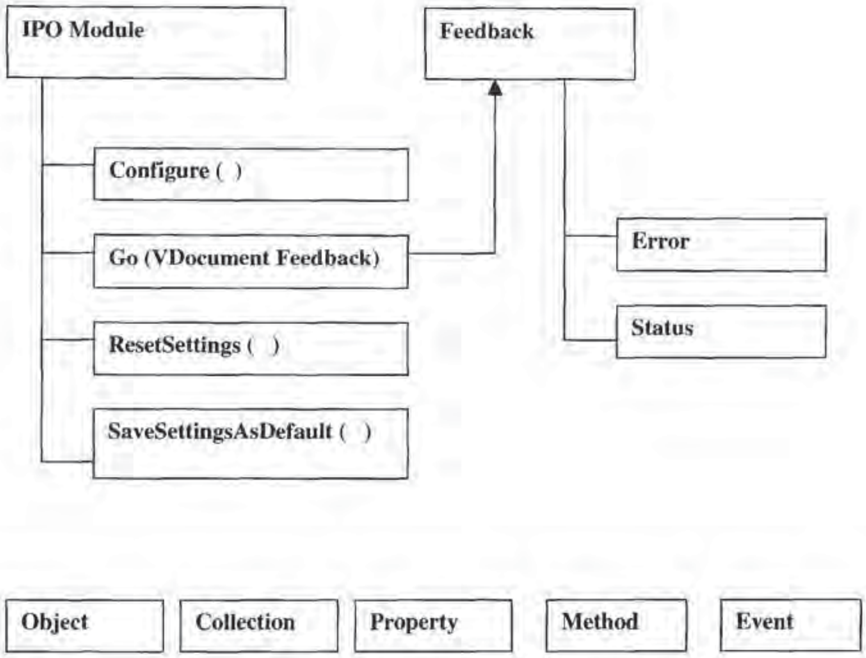

IPO Module
Configure ( )
Go (VDocument Feedback)
ResetSettings ( )
SaveSettingsAsDefault ( )
Feedback
Error
Status
Object
Collection
Property
Method
Event

FIG. 45

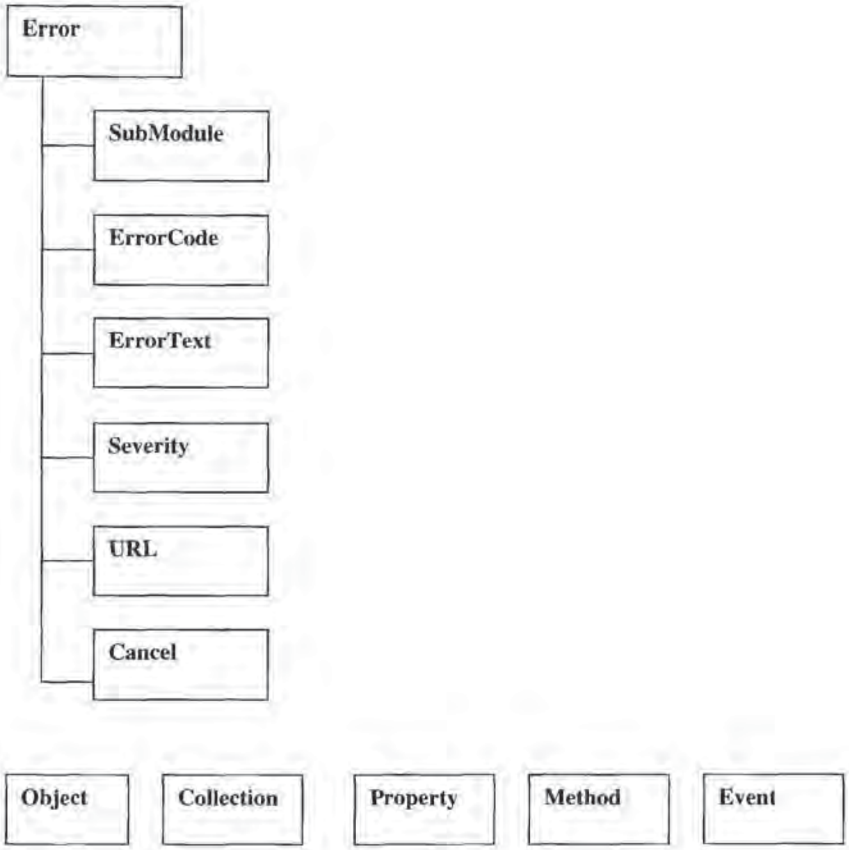

Error
SubModule
ErrorCode
ErrorText
Severity
URL
Cancel
Object
Collection
Property
Method
Event

FIG. 46

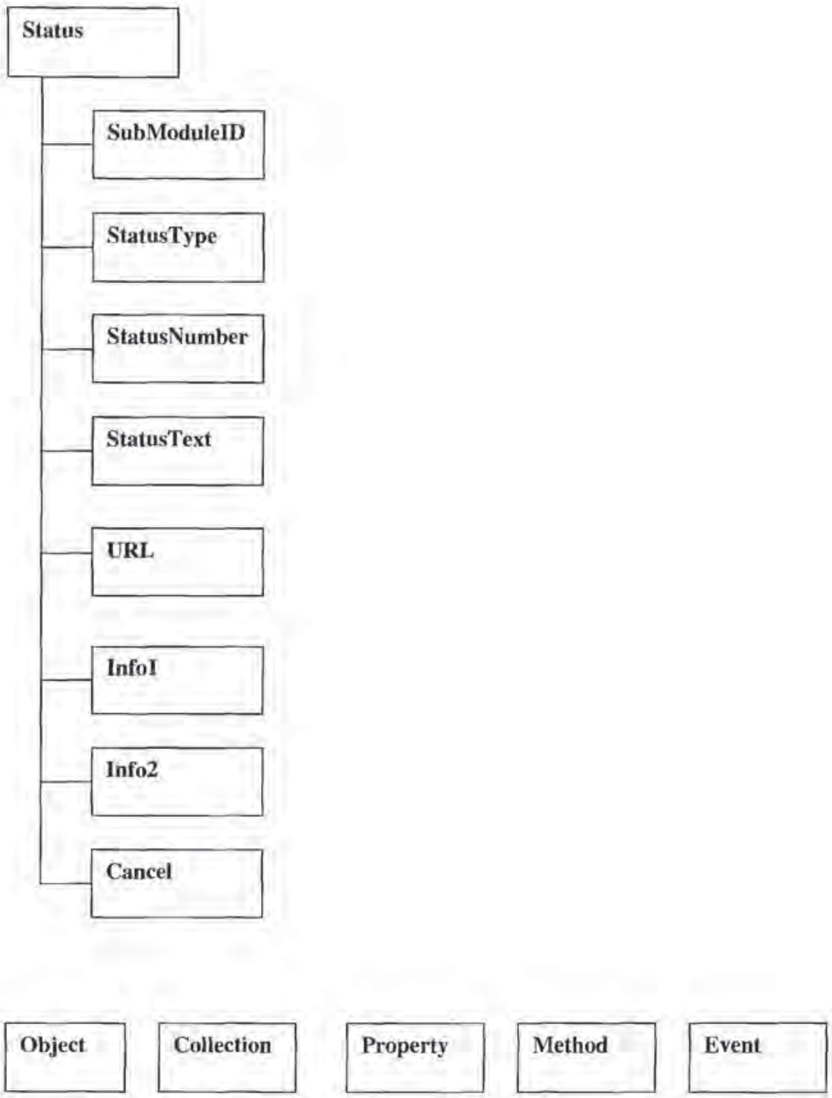

Status
SubModuleID
StatusType
StatusNumber
StatusText
URL
InfoI
Info2
Cancel
Object
Collection
Property
Method
Event

FIG. 47

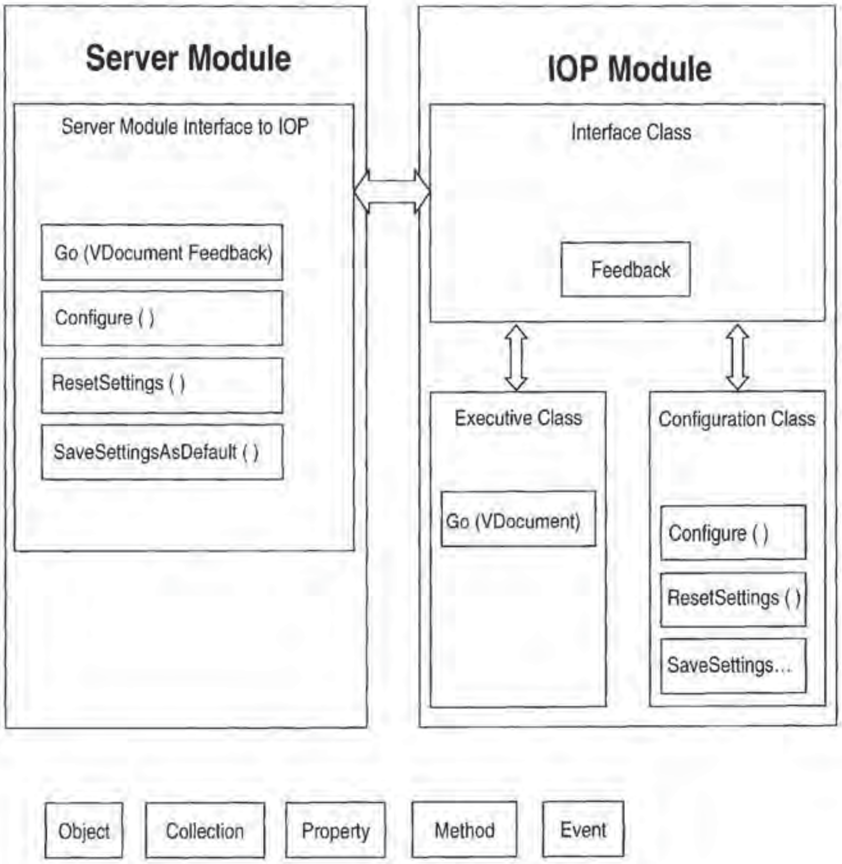

Server Module
Server Module Interface to IOP
Go (VDocument Feedback)
Configure ( )
ResetSettings ( )
SaveSettingsAsDefault ( )
IOP Module
Interface Class
Feedback
Executive Class
Go (VDocument)
Configuration Class
Configure ( )
ResetSettings ( )
SaveSettings...
Object
Collection
Property
Method
Event

FIG. 48

DISTRIBUTED COMPUTER ARCHITECTURE AND PROCESS FOR VIRTUAL COPYING

RELATED APPLICATIONS

This application claims priority to U.S. Provisional Application 60/108,798, filed Nov. 13, 1998, incorporated herein by reference.

This application is related to, a continuation-in-part application of, and claims priority to, the following non-provisional applications: Ser. No. 08/950,838, filed Oct. 15, 1997, now U.S. Pat. No. 6,185,590;

Ser. No. 08/911,083, filed Aug. 14, 1997, now abandoned;

Ser. No. 08/950,911, filed Oct. 15, 1997, now abandoned;

Ser. No. 08/950,837, filed Oct. 15, 1997, now abandoned;

Ser. No. 08/950,738, filed Oct. 15, 1997, now abandoned;

Ser. No. 08/950,741, filed Oct. 15, 1997, now abandoned; all of which are hereby incorporated by reference.

This application is related to, and claims priority to, the following provisional applications by way the claim of priority of the above listed non-provisional applications:

Oct. 18, 1996, Ser. No. 60/028,129;

Oct. 18, 1996, Ser. No. 60/028,522;

Oct. 18, 1996, Ser. No. 60/028,128;

Oct. 18, 1996, Ser. No. 60/028,697;

Oct. 18, 1996, Ser. No. 60/028,639;

Oct. 18, 1996, Ser. No. 60/028,685; all of which are hereby incorporated by reference.

FIELD OF THE INVENTION

The present invention is generally related to a computer architecture and process for stand-alone and/or distributed environment, and more particularly to a computer architecture and process using a substantially uniform management in a stand-alone and/or distributed computing environment including, for example, client server and/or intranet and/or internet operating environments.

BACKGROUND OF THE RELATED ART

A "C" or "C++"- Level API (hereinafter "C" Level), which is the native language and interface for a vast repository of core technologies from small software vendors and research laboratories, are unique to each designer. The designer of a text retrieval "C"-API will generally implement an interface that is completely different than a second inventor creating a "C"-level API for OCR.

Every "C"-level API is unique, both in its choice of API syntax as well as its method for implementing the syntax. Some API's consist of one or two functions that take parameters offering options for different features offered by the technology. Other APIs consist of hundreds of functions with few arguments, where each function is associated with a particular feature of the core technology. Other APIs provide a mixture of some features being combined with one function with many arguments, while other features are separated into individual function calls.

Without any constraints, each designer of a core technology chooses to implement his or her technology with an interface that is suitable to the subject or simply was the most expedient choice of the moment. Since there are no constraints, a "IC"-level API has a totally unpredictable interface that can often be the hindrance to using the core technology.

Additionally, every API manages errors differently further complicating the problems described above. Some APIs return a consistent error code for each function. Error management in this case is very organized and manageable. Other APIs return error codes as one of the parameters passed to the function. There are APIs that mix the choice of error management and have some functions return an error code while other functions pass the error code as a parameter of a function. Errors can also be managed by a callback function, eliminating the need for passing any error code as part of the function. In some instances of a poorly implemented API the errors are not passed back at all.

Every engine, such as a text retrieval or an OCR (Optical Character Recognition) engine, has a unique interface. This interface is generally a "C"-level API (Application Program Interface). Further, an API can at any time be synchronous, asynchronous, manage one or more callbacks, require input, pass back output, carry a variety of different styles of functions, return values or not return values, and implement the unpredictable. This unpredictability in APIs further compounds the problem of developing a sane way of interfacing between components and APIs.

To date, because of the complexities of "C"-level APIs and components interfacing thereto, the only way to create a component out of an existing "C"-level API is to have an experienced programmer in the field to do the work. Humans can intelligently analyze an API, and create a component based on intelligent decisions and experiences. In most cases, the learning curve for understanding and integrating a new engine can be one man-month to several man-years and generally requires highly experienced "IC" programmers. Requiring a human to perform the necessary work is costly, and subject to real-life human constraints.

Since there is no structure or format for implementing "C"-level APIs, the ability to automatically transform a unique API into a standard component would seem impossible, since that would take a nearly-human level of intelligence.

In addition, in spite of the continued automation of business processes, companies are increasing their paper use by 25–30% and spending up to 15% of their total budget on managing paper. Companies are often running dual processes—a computerized process along with the corresponding paper filing system—and paying an extraordinary price for it. Just a few examples will illustrate the problem: 1) accounting clerks are maintaining paper invoices with information that is also being re-keyed into accounting systems, 2) administrative assistants are filing incoming correspondence in cabinets for customers whose records are also being electronically maintained by contact management systems, 3) help desk operators are storing complaints sent in on paper while also tracking those complaints in a computerized system. Additional industry trends include the following:

For every $100M in increased revenues, a company will use 8.8 million additional pages of paper

The Document Management market is expected to grow at 30% per year

The digital device market is growing at 20% per year

Estimates show the web-based document imaging market growing at 50% per year

The digital device manufacturers, especially the copier companies, are heavily promoting the ability to connect their devices to networks, but have not been able to deliver an effective software solution to date.

Businesses continue to automate more processes, but managing the associated paper is often ignored, resulting in inefficiency and higher costs.

I have determined that a component factory, if it is to be truly automated or manually expedited, must be able to take any "C"-level API and transform it into a component.

I have also determined an efficient and workable design for an architecture to define the migration path for any "C"-level API into a component.

I have also determined that it is desirable to develop software tools for automatically generating reusable software components from core software technologies, thus making these software technologies available to a much larger user base.

I have further determined that it is desirable to design a distributed computer architecture and process for manually and/or automatically generating reusable software components. The computer architecture may be implemented using a client server and/or intranet and/or internet operating environments.

I have further determined that it is desirable to design a computer architecture and process for image viewing in a stand-alone and/or distributed environment. The computer architecture and process optionally uses a substantially uniform management layer in a stand-alone and/or distributed computing environment including, for example, client server and/or intranet and/or internet operating environments.

I have further determined that it is desirable to enable a typical PC user to add electronic paper processing to their existing business process.

I have further determined that it is desirable to enable software that manages paper so that it can be electronically and seamlessly copied in and out of devices and business applications (such as Microsoft Office, Microsoft Exchange, Lotus Notes) with an optional single-step Go operation.

SUMMARY OF THE INVENTION

One would expect the translating a "C"-level API from its native state into a component would require human-level intelligence. This is mainly because "C"-level APIs have virtually no constraints as to how they can be implemented. This means that there are an infinity variations of APIs, which can only be managed by human-level intelligence. While this point is true, I have determined that the appropriate solution starts at the other side of the equation, which is the component itself.

My solution starts out with a definition of a component that can sustain the feature/function requirements of any API. In other words, the interface of a generic component can be defined such that the features and functions of virtually any API can be re-implemented within its bounds. The two known end-points are, for example, the "C"-level API that generally starts with each component (although other programming languages may also be used and are within the scope of the present invention), and the component interface that represents any set of features/functions on the other side. The component factory migrates the original "C"-level API from its original state into the generic interface defined by the topmost layer. The first feature that can be demonstrated is that there is a topmost layer that can define a component interface that can represent the features/functions of most core technologies.

The component factory migrates the "C"-level API to the topmost level. Doing this in one large step would be impossible since the "C"-level API has a near-infinite variety of styles. However, the architecture advantageously has enough well-defined and well-structured layers for implementing the topmost component interface, for creating the component factory.

The computer architecture is designed for managing a diverse set of independent core technologies ("engines") using a single consistent framework. The architecture balances two seemingly opposing requirements: the need to provide a single consistent interface to many different engines with the ability to access the unique features of each engine.

The benefit of the architecture is that it enables a company to rapidly "wrap" a sophisticated technology so that other high-level developers can easily learn and implement the core technology. The computer architecture is therefore a middleware or enabling technology.

Another benefit of the architecture is that it provides a high-level specification for a consistent interface to any core technology. Once a high-level developer learns the interface described herein for one engine, that knowledge is easily transferable to other engines that are implemented using the architecture. For example, once a high-level developer learns to use the computer architecture for OCR (Optical Character Recognition), using the computer architecture for other engines, such as barcode recognition or forms processing, is trivial.

The architecture described herein is, at once, a framework for rapidly wrapping sophisticated technologies into high-level components, as well as a framework for high-level developers to communicate with a diverse set of engines. The creating of a component factory is based on the fact that the architecture defines a clear path for "wrapping" any C-level API into a component using simple structures and many rote steps. This process is currently being done in an inefficient manner by a programmer in the field.

In addition, the method described herein for creating a component factory creates a well-defined multi-tiered architecture for a component and automates, substantially automates, or manually expedites hereinafter automate the process of migrating a "C"-API from its native state through the various tiers of the architecture resulting in a standardized component. Advantageously, the method described herein does not base the component factory on making human-level intelligent decisions on how to translate a "C"-API into a component. Rather, by creating a well-defined architecture described below that is multi-tiered, the method is a series of incremental steps that need to be taken to migrate the "C"-API from one tier within the architecture to the next. In this way each incremental step is not a major one, but in sequence the entire series of steps will result in a component.

Since each step of migration is not a major one, the chances for automating these steps is significantly higher and the likelihood of being able to create the component factory becomes feasible. This approach is in fact what makes the method cost-effective, since the alternative approach, i.e., computer-generated human-level decision making, has many years before becoming sophisticated enough to replace humans in any realistic decision-making process.

The main features of the architecture are twofold:

1) Defining system architecture that describes in detail how to implement a component from a "C"-level API;
2) Creating a component factory by automating the migration of a "C"-level API from one tier within the architecture to the next.

The latter feature is the key to actually making the component factory feasible. With a fixed architecture that can be used to implement a "C"-level API as a component (using a programmer), that same architecture can be used as the basis for the component factory model.

In order to make the component factory, each step of the architecture needs to be designed to facilitate automation or manually expedited. In other words, I have determined that automating/expediting the process of taking the original "C"-level API and migrating it to a Level 1 layer, and then a Level 1 to a Level 2, and then a Level 2 to a Level 3 layer, and so on, the component has been implemented automatically or more efficiently. The component factory is therefore a sum of the ability to automate migrating the "C"-level API from one layer to the next within a well-defined architecture for implementing components.

There are numerous core technologies, such as text-retrieval and ICR (Intelligent Character Recognition), that have already been implemented, and are only available as "C"-level APIs. Many, if not most, core technologies are first released exclusively as "C"-level APIs. While there are integrators and corporations who have the team of technologists who can integrate these "C"-level APIs in-house, most companies are looking for component versions that can be implemented at a much higher level.

Therefore, many of the core technologies that are only available in a "C"-level API are not being used due to their inaccessible interface. The benefit of the component factory is that it can rapidly make available core technologies implemented as "C" APIs that would otherwise be underutilized or dormant in research labs by converting them to high-level components that can be used by millions of power-PC users.

With the advent of the World Wide Web (WEB) this opportunity has increased exponentially. The WEB is now home to a vast number of WEB authors with minimal formal training who can implement HTML pages and build web sites. One of the fundamental technologies for extending the capability of the WEB from simple page viewing to interactive and sophisticated applications is components.

A component extends the capability of HTML by enabling a WEB author to add core technology as a pre-packaged technology. Since components are fundamental to the growth and usability of the WEB, having a component factory that can translate "C"-level toolkits into components that are then usable within WEB sites opens a vast and new worldwide market to these technologies.

The purpose of the Virtual Copier ("VC") aspect of the present invention is to enable a typical PC user to add electronic paper processing to their existing business process. VC is an extension of the concept we understand as copying. In its simplest form it extends the notion of copying from a process that involves paper going through a conventional copier device, to a process that involves paper being scanned from a device at one location and copied to a device at another location. In its more sophisticated form, VC can copy paper from a device at one location directly into a business application residing on a network or on the Internet, or visa versa. The VC invention is software that manages paper so that it can be electronically and seamlessly copied in and out of devices and business applications (such as Microsoft Office, Microsoft Exchange, Lotus Notes) with an optional single-step Go operation. The VC software can reside on a PC, LAN/WAN server, digital device (such as a digital copier), or on a web server to be accessed over the Internet.

Virtual Copier is designed to solve the corporate paper problem by enabling existing web-based and client-server applications to manage paper as part of their solution. Virtual Copier links the familiar and universal world of paper and digital devices to web-based and client-server applications. The result is that the automated business processes become the primary storage of paper in electronic form. Information that is typically managed and processed in paper form is "copied" into the system and managed by the business processes with which users are accustomed, which is made possible by using Virtual Copier. Simple extensions of Virtual Copier support seamless electronic outsourcing of paper processing and archival services over the web.

Virtual Copier is a unique combination of an intuitive application built on an open component architecture that delivers a simple innovation: provide paper processing to existing Intranet and client-server business processes without any fuss. Whether it is an office clerk that needs to easily copy a report from a desktop scanner to the company's Intranet-networked copier, or an accounting software integrator that wants to embed paper processing, Virtual Copier offers a simple solution. To the office clerk Virtual Copier is a document imaging application packaged in the familiar setting of an office copier. To the integrator, the underlying open architecture of Virtual Copier offers a simple integration path for embedding paper processing into its client-server or web-based software solution.

Although managing paper manually is one of the great problems facing corporations, there has been little innovation in enabling those workers to eliminate the need to continuously work with paper manually. Much of the problem stems from the complexity of traditional document management systems, which require days of training and months to become familiar with the system in order to be proficient. Virtual Copier was designed to be as simple as a copier to operate, and yet still provide the complete capability of integrating paper with existing business applications. By simplifying the interface and underlying software infrastructure, VC can manage paper in electronic form as easily as is currently done in physical form.

VC extends the notion of a copier, which simply replicates the image of an original document onto another piece of paper using a single GO or START button, to do a similar operation in software so that the image gets seamlessly replicated into other devices or applications or the Internet.

An example of this is the actual implementation of Virtual Copier as a consumer product. The interface of the consumer product called Virtual Copier has a Go button much like a physical copier. This GO button can copy paper, whether physical or electronic, from one device and or application to another device and/or application.

What makes Virtual Copier as simple as its physical counterpart in at least one embodiment is the fact that it replicates the identical motions that a user who is making a copy using a physical photocopier goes through. When a user photocopies a document, he/she selects where they want to copy from (i.e. the sheet feeder), where the user wants to copy to (i.e. 6 copies collated and stapled) and then presses a GO button to actually carry out the photocopy process. With Virtual Copier the process feels familiar because the sequence is the same with just the Power VC portion of the main Virtual Copier window.

The power of Virtual Copier is the fact that the From can be a physical device (e.g. digital copier, fax or scanner) or an application (e.g. Lotus Notes, Microsoft Exchange, the Internet, or an electronic filing system). The To can also be a physical device (e.g. a fax, digital copier, or printer) or an application (e.g. Lotus Notes, Microsoft Exchange, the Internet, or an electronic filing system). Even though paper is being copied electronically from devices to applications, from applications to devices, from devices to devices, or from applications to applications, the user simply has one sequence to execute: select From, select To, and then press GO. Virtual Copier will accomplish all translations between device and applications automatically and seamlessly.

Another reason that paper is still a major corporate issue is that traditional document management systems require that a company invest in a whole new system just to store electronic images. Although this is the only way that document management systems have been designed and delivered, it is in fact highly inefficient. Most companies already manage information about physical documents in some form of software applications.

For example, accounting systems have long been used to maintain information about invoices and bills that arrive into a company from outside sources as physical pieces of paper. When an invoice arrives, its information is keyed into the accounting software, where balances are maintained and accounts payable information is coordinated. Yet the original invoice is stored manually, and every time that a request is made for a copy of the signed invoice, someone manually retrieves the invoice from a physical filing cabinet. Accounting systems, like most business applications, typically have no way of maintaining an electronic copy of the physical invoice, and adding a document management system to an accounting system is cumbersome, costly, and difficult to maintain, and even more difficult to coordinate.

Virtual Copier solves this problem in at least one embodiment by copying paper directly into the existing accounting system. Simply adding a To item in the Virtual Copier window enables a user to copy paper directly into the appropriate accounting record of the existing accounting system. This requires no retraining (users who are trained on the accounting system will still use the accounting system in the same way), requires no document management system (the electronic copy of the document is actually being maintained by the accounting system itself), there is no coordination between two systems (Virtual Copier embeds the invoice with the appropriate accounting record), and it is simple (one Go button).

What is true with regard to the example above of an accounting system is true of most other business applications. The power of Virtual Copier is that it can turn an information system into a document management system by adding support for electronic paper directly into the existing business application, whether it is a client, server-based, or web-based system.

Virtual Copier enables corporations to perform sophisticated document imaging with their existing Web-based and client-server applications through a user interface that is as familiar as the office copier. Virtual Copier can be used out-of-the-box as a standalone application to copy, scan, fax, or print images using existing digital devices within corporate environments or across the web. With the extensions, as described below, Virtual Copier can be integrated into Web-based and client server applications, such as ERP or accounting systems, to eliminate paper from existing business processes and legacy applications. Virtual Copier can also be used to support seamless access to document image processing and archival over the web since, in at least one embodiment, the VC interface is implemented as a software application.

VC is architected as an application that delivers end-user functionality while remaining open to third-parties extensions. For example, VC can be viewed as a copier. Like a copier, VC takes paper in, and produces paper going out. The only difference is that VC does not distinguish between electronic and physical paper.

To accommodate third-party extensions, VC is divided into five essential modules. Each module is a counterpart to an aspect that is found on a conventional copier. Based on the modular design of VC, each aspect of VC can be independently extended, offering much greater flexibility than conventional copiers.

The five core modules of VC are:

Input Module—The Input Module manages paper or electronic paper entering VC. This module manages imaging devices to input paper through scanners, MFPs, or the new breed of digital copiers. The Input Module also manages reading electronic paper from third-party or proprietary applications. The counterpart to VC's Input Module on a conventional copier is the scanner subsystem.

Output Module—The Output Module manages paper or electronic paper exiting VC. Like the Input Module, this module manages imaging devices to output paper to standard Windows printers, specialty image printers, MFPs, or the new breed of digital copiers. The Output Module also manages writing electronic paper to third-party or proprietary applications. The counterpart to VC's Output Module on a conventional copier is the printer or fax subsystem.

Process Module—The Process Module applies processing to the electronic paper as it is being copied. Examples of a process are OCR and ICR. The Process Module can also apply non-imaging functionality as well, such as workflow or other relevant tie-ins to the electronic paper as it is being copied. One of the advantages of VC over conventional copiers is that multiple processes can be applied to a single virtual copy. The counterpart to VC's Process Module on a conventional copier is the controller.

Client Module—The Client Module presents the electronic paper as it is being copied, and any relevant information related to the input or output functions. For example, if the Output Module is directed to a printer, then the Client Module might present the finishing capabilities; if the Output Module is directed to Goldmine, then the Client Module might present the target contact record to which the document is being copied. The counterpart to VC's Client Module on a conventional copier is the panel.

Server Module—Unlike conventional copiers, VC's Server Module is a unique subsystem that can communicate with the other modules as well as third-party applications. The Server Module is what makes VC a far more powerful concept than simply an application that can control a scanner and a printer to mimic a copier. The Server Module can be used to combine third-party applications with the new breed of digital imaging devices to create unique and custom virtual copier solutions. A virtual copier can be created with VC by combining a scanner with a printer; or by combining a scanner with an application; or by combing an application with an image printer. In each case VC is dynamically creating a custom virtual copier, with a complete understanding of how paper flows from the source to its destination. There is no counterpart to VC's Server Module on a conventional copier.

One of the primary design goals of VC is to make it simple to integrate VC with third-party applications. There are two options to integrating VC into a third-party application: running VC as an external service, or embedding VC as an underlying service.

VC is in one embodiment and optionally a standalone application that enables a user to scan (copy) paper from a device to a third-party application, and to print (copy) the reference of an image document from a third-party application to a printing device. VC does not require the third-party application to be aware that VC is operating. Rather, VC recognizes that the third-party application is running, and it intelligently copies paper to and from that application.

In this scenario the user is interacting with VC's Client Module in order to execute a copy operation to and from the third-party application. There does not have to be any changes made to the third-party application, not even to its interface, in order for VC to operate. The user of VC only knows that to copy to and from the third-party application, a custom Input and Output Module must be selected, and the Go button is pressed.

In order to support copying to and from a third-party application, VC must be able to support extensions that understand each third-party application. This is accomplished through the Input and Output Modules. The Client, Server, and even Process Modules remain independent across third-party applications. However, in order to support outputting to a third-party application, an Output Module is developed that is unique to that third-party application. Likewise, an Input Module is developed that is unique to a third-party application in order to support reading images from that application.

It is the optional Input and Output Modules that render VC extendable. For each third-party application there is a unique pair of Input and Output Modules that understand the third-party application, and how to copy images to and from that application. Each Input and Output Module registers itself to the Windows registry so that the Server Module knows how to find them. In this way Virtual Copier can grow indefinitely, to support any number of third-party applications.

The significant point is that the Input and Output Modules have their own interface, and can be developed independently from any other module. As long as the Input and Output Module conform to the API specified in this document it will plug-and-play with VC. VC will be able to mix and match the custom Input and Output Module with its standard and other custom Input and Output Modules.

A third-party application can also use the services of VC without its user interface. That is, a third-party application can embed VC's functionality and provide its own interface to its functionality. For example, rather than have VC as a separate application, a special button can be placed on a third-party application that launches VC in the background.

VC is designed so that the Server Module can run independently from the Client Module. All the core functionality, including communicating with the Input, Output, and Process Modules, are performed directly by the Server Module. The Client Module is generally simply an interface to the Server Module. Therefore, all the services of the Server Module can be made available in the background to a third-party application without the need for an interface. The third-party application can in fact become the user's interface to VC.

In order to support VC operating in the background a third-party application merely has to communicate with the Server Module directly, as described later in this document. The Server Module, as all modules in VC, support COM-based interfaces for simple and direct support from all major Windows development environments.

Accordingly, it is a feature and advantage of the present invention to implement a component factory, that is automated or manually expedited.

It is another feature and advantage of the present invention to be able to take any "C"-level API and transform it into a component.

It is another feature and advantage of the present invention to define an efficient and workable design for an architecture to provide the migration path for any C-level API into a component.

It is another feature and advantage of the present invention to develop software tools for automatically generating reusable software components from core software technologies.

It is another feature and advantage of the present invention to develop software tools to make software components available to a much larger user base.

It is another feature and advantage of the present invention in providing a distributed computer architecture and process for manually and/or automatically generating reusable software components.

It is another feature and advantage of the present invention in providing a distributed computer architecture and process for manually and/or automatically generating reusable software components where the computer architecture is implemented using a client server and/or intranet and/or internet operating environments.

It is another feature and advantage of the present invention in providing a computer architecture and process for image viewing in a stand-alone and/or distributed environment.

It is another feature and advantage of the present invention in providing a computer architecture and process that uses a substantially uniform management layer in a stand-alone and/or distributed computing environment including, for example, client server and/or intranet and/or internet operating environments.

It is another feature and advantage of the present invention to enable a typical PC user to add electronic paper processing to their existing business process.

It is another feature and advantage of the present invention to enable software that manages paper so that it can be electronically and seamlessly copied in and out of devices and business applications (such as Microsoft Office, Microsoft Exchange, Lotus Notes) with an optional single-step Go operation.

The present invention is based, in part, on my discovery that it is possible to make the component factory, and that each step of the architecture is designed to facilitate automation or manually design of components. The present invention is also based, in part, on my discovery that by automating/expediting the process of taking the original "C"-level API and migrating it to a Level 1 layer, and then a Level 1 to a Level 2, and then a Level 2 to a Level 3 layer, and so on, the component has been implemented automatically and/or more manually efficiently. The component factory is therefore a sum of the ability to automate migrating the "C"-level API from one layer to the next within a well-defined architecture for implementing components.

The present invention is also based, in part, on my discovery that the object manager and engine object component layers may be advantageously be designed to operate independently, thereby making possible a distributed computing environment, as described below in detail. I have further discovered that an efficient method of implementing the engine object component layer is by using pre-populated tables/files. I have further discovered that the engine management layer may be advantageously divided into a three layer structure of load/unload engine, dynamic linking engine function calls, and initialize engine setting.

In accordance with one embodiment of the invention, a computer implemented process migrates a program specific Application Programmer Interface (API) from an original state into a generic interface by building an object for each engine. The object provides substantially uniform access to the engine and engine settings associated with the engine. The computer implemented process includes the step of providing an engine management function interfacing with the program specific API. The engine management function furnishes a protective wrapper for each function call associated with the engine, trapping errors, and provides error management and administration to prevent conditions associated with improper engine functioning. The process optionally includes the step of providing an engine configuration function transforming API calls received from the program specific API into standardized calls. The engine configuration function provides additional functionality, including safely loading and unloading the engine. The process optionally includes the step of providing an engine function managing the standardized calls for each engine, thereby providing substantially uniform access to the engine and the engine settings associated with the engine.

In accordance with another embodiment of the invention, a computer implemented method migrates at least one program specific Application Programmer Interface (API) from an original state into a generic interface by building an object for each engine. The object provides substantially uniform access to the engine and engine settings associated with the engine. The computer implemented method includes the steps of defining a substantially consistent interface for individual object components that represent diverse technologies, and migrating a plurality of engines to the consistent interface. The computer implemented method also includes the step of substantially automatically and/or substantially uniformly, managing the individual object components using a predefined object manager and the consistent interface.

In accordance with another embodiment of the invention, a computer architecture migrates at least one program specific Application Programmer Interface (API) from an original state into a generic interface by building an object for each engine. The object provides substantially uniform access to the engine and engine settings associated with the engine. The computer architecture includes an engine management layer interfacing with the program specific API and providing engine management and administration, an engine configuration layer transforming API calls received from the program specific API into standardized calls, and an engine layer managing the standardized calls for each engine.

In accordance with another embodiment of the invention, an engine management layer configures a computer architecture to perform one or more computer implemented or computer assisted operations. The computer operations include one or more of loading and unloading engine dynamic link libraries into and out of memory for each engine, mapping at least one engine function to at least one corresponding engine object, providing general error detection and error correction for each engine, determining and matching arguments and returning values for mapping the at least one engine function to the at least one corresponding engine object, and/or managing error feedback from the at least one program specific API.

In accordance with another embodiment of the invention, a distributed computer system migrates a program specific Application Programmer Interface (API) from an original state into a generic interface by building an object for each engine. The object provides substantially uniform access to the engine and engine settings associated with the engine. The distributed computer system includes a server configured to include at least one engine having an engine interface

providing one or more features to be executed, and at least one engine component configured to execute the one or more features of the engine by mapping a substantially consistent interface to the engine interface of the engine. The distributed computer system also includes at least one client configured to be connectable to the server and optionally configured to be connectable to another server.

The client includes an object manager layer communicable with and managing the at least one engine component stored on the server via the substantially consistent interface.

In accordance with another embodiment of the invention, a distributed computer implemented process migrates a program specific Application Programmer Interface (API) from an original state into a generic interface by building an object for each engine. The object provides substantially uniform access to the engine and engine settings associated with the engine. The computer implemented process includes the step of providing, on a server, at least one engine having an engine interface, and providing one or more features to be executed. The computer implemented process also includes the step of providing, on at least one of the server and another server connectable to the server, at least one engine component configured to execute the one or more features of the engine by mapping a substantially consistent interface to the engine interface of the engine. The computer implemented process also includes the step of providing, on a client configured to be connectable to the server and optionally configured to be connectable to the another server, an object manager layer communicable with and managing the at least one engine component via the substantially consistent interface.

In accordance with another embodiment of the invention, an image viewer process views at least one document image including an electronic document image, and performs viewing operations to the electronic document image. The process includes the step of selecting, by the user, one of a plurality of image viewing perspectives. Each of the plurality of image viewing perspectives provide the user the capability of viewing the document image in accordance with a different predefined user perspective. The process also includes the steps of selecting, by the user, using the image viewer process the document image to be viewed, and retrieving, by the image viewer process, the document image. The process also includes the step of displaying, by the image viewer process, the selected document image in accordance with an image viewing perspective selected by the user.

In accordance with another embodiment of the invention, a computer readable tangible medium is provided that stores the process thereon, for execution by the computer.

A computer data management system includes at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications. The computer data management system is responsively connectable at least one of locally and via the Internet, and includes at least one memory storing a plurality of interface protocols for interfacing and communicating, and at least one processor responsively connectable to the at least one memory. The processor implements the plurality of interface protocols as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, the external devices and applications include, for example, a printer, a facsimile, and a scanner. In one embodiment, the computer data management system includes the capability to integrate an image using software so that the image gets seamlessly replicated and transmitted to at least one of other devices and applications, and via the Internet. In one embodiment, the computer data management system includes the capability to integrate the electronic images into a destination application without the need to modify the destination application.

In one embodiment, the computer data management system includes an interface that enables copying images between physical devices, applications, and the Internet using a single "GO" operation. In one embodiment, the computer data management system includes the capability of adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, the software application includes at least one input module managing data comprising at least one of paper and electronic paper input to the computer data management system, and managing at least one imaging device to input the data through at least one of a scanner and a digital copier, and managing the electronic paper from at least one third-party software applications; at least one output module managing the data output from the computer data management system, managing at least one imaging device to output the data to at least one of a standard Windows printer, an image printer, and a digital copier, and managing the output of the data to the third-party software application; at least one process module applying at least one data processing to the data comprising the at least one of the paper and the electronic paper as it is being copied, applying additional functionality including at least one of workflow and processing functionality to the data comprising the at least one of paper and electronic paper as it is being copied, and applying multiple processes to a single virtual copy; at least one client module presenting the data comprising the at least one of paper and electronic paper as it is being copied, and information related to at least one of the input and output functions; and at least one server module communicable with said at least one input, output, client, and process modules and external applications, and capable of dynamically combining the external applications with at least one of digital capturing devices and digital imaging devices.

In one embodiment, one or more of the external devices and applications integrates the computer data management system into an external application via one of running the computer data management system, as an external service and embedding the computer data management system as an embedded service.

In one embodiment, the server module includes enable virtual copy operation means for initiating, canceling, and resetting said computer data management system; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer data management system copy operation in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, the server module includes at least one server module application programmer interface (API).

IN one mebodiment, the server module application programmer interface (API) comprises the COM-based interfaces: at least one modules object maintaining a first list of available input, output, and process modules; at least one program object maintaining a second list of currently selected input, output, and process modules; at least one document object maintaining information regarding a current document being copied; at least one system management method object used to initiate, cancel, and reset said computer data management system; and at least one system management event object used to provide feedback to the Client Module.

In one embodiment, a computer data management system includes at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The computer data management system comprises: a first capability to integrate an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; a second capability to integrate electronic images into existing applications without the need to modify the destination application; an interface comprising a software application that enables copying images between physical devices, applications, and the Internet using a single "GO" operation; and a third capability of adding at least one of electronic document and paper processing with a single programming step.

A computer data management system capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications at least one of locally and via the Internet. The computer data management system includes at least one memory storing at least one of a common and universal interface protocol for interfacing and communicating; and at least one processor responsively connectable to said at least one memory, and implementing the at least one common and universal interface protocol as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, a computer readable tangible medium stores instructions for implementing a process driven by a computer implemented on at least one of an electronic image, graphics and document management system capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including at least one of an external device and application at least one of locally and via the Internet. The instructions control the computer to perform the process of: storing at least one of a common and universal interface protocol for interfacing and communicating in at least one memory; and implementing the at least one of common and universal interface protocol as a software application via at least one processor for interfacing and communicating with the plurality of external destinations including the at least one external device and application.

In one embodiment, a computer data management system includes at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The computer data management system includes a single function copy operation linking devices, applications and the Internet including at least one a go operation, a single function paper copy between devices and software applications, and a single function paper copy between software applications and devices; a one step programming method to add paper support to electronic business processes including at least one of a one step method of supporting paper within electronic business process application optionally including legacy systems with no or minimal reprogramming of the electronic business process application, a method of recreating a module oriented copier in software; and a copier interface implemented as software application including at least one of a virtual copier interface method of presenting to a user an operation of at least one of copying files and electronic images, at least one of to and from, at least one of digital imaging devices and software applications, in a substantially single step, and presenting users with direct access to at least one of tutorial and options from a main application window.

In one embodiment, a server module includes enable virtual copy operation means for initiating, canceling, and resetting said computer data management system; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer data management system copy operation in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, a computer data management method includes at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The method comprises the steps of integrating an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; integrating electronic images into existing applications without the need to modify the destination application; interfacing via a software application enabling copying images between physical devices, applications, and the Internet using a single "GO" operation; and adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, a server method includes initiating, canceling, and resetting said computer data management system; maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintaining said input, output, and process modules currently being used for a current computer data management system copy operation in a program object, and saving the currently active modules in a process template file; and maintaining information regarding a current file being copied, and saving the information in a document template file.

A computer data administration system includes at least one of an electronic image, graphics and document administration system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications. The computer data administration system is responsively connectable at least one of locally and via the Internet, and includes at least one memory storing a plurality of interface protocols for interfacing and communicating, and at least one processor responsively connectable to the at least one memory. The processor implements the plurality of interface protocols as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, the external devices and applications include, for example, a printer, a facsimile, and a scanner. In one embodiment, the computer data administration system includes the capability to integrate an image using software so that the image gets seamlessly replicated and transmitted to at least one of other devices and applications, and via the Internet. In one embodiment, the computer data administration system includes the capability to integrate the electronic images into a destination application without the need to modify the destination application.

In one embodiment, the computer data administration system includes an interface that enables copying images between physical devices, applications, and the Internet using a single "GO" operation. In one embodiment, the computer data administration system includes the capability of adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, the software application includes at least one input module managing data comprising at least one of paper and electronic paper input to the computer data administration system, and managing at least one imaging device to input the data through at least one of a scanner and a digital copier, and managing the electronic paper from at least one third-party software applications; at least one output module managing the data output from the computer data administration system, managing at least one imaging device to output the data to at least one of a standard Windows printer, an image printer, and a digital copier, and managing the output of the data to the third-party software application; at least one process module applying at least one data processing to the data comprising the at least one of the paper and the electronic paper as it is being copied, applying additional functionality including at least one of workflow and processing functionality to the data comprising the at least one of paper and electronic paper as it is being copied, and applying multiple processes to a single virtual copy; at least one client module presenting the data comprising the at least one of paper and electronic paper as it is being copied, and information related to at least one of the input and output functions; and at least one server module communicable with said at least one input, output, client, and process modules and external applications, and capable of dynamically combining the external applications with at least one of digital capturing devices and digital imaging devices.

In one embodiment, one or more of the external devices and applications integrates the computer data administration system into an external application via one of running the computer data administration system, as an external service and embedding the computer data administration system as an embedded service.

In one embodiment, the server module includes enable virtual copy operation means for initiating, canceling, and resetting said computer data administration system; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data administration system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer data administration system copy operation in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, the server module includes at least one server module application programmer interface (API). IN one mebodiment, the server module application programmer interface (API) comprises the COM-based interfaces: at least one modules object maintaining a first list of available input, output, and process modules; at least one program object maintaining a second list of currently selected input, output, and process modules; at least one document object maintaining information regarding a current document being copied; at least one system administration method object used to initiate, cancel, and reset said computer data administration system; and at least one system administration event object used to provide feedback to the Client Module.

In one embodiment, a computer data administration system includes at least one of an electronic image, graphics and document administration system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The computer data administration system comprises: a first capability to integrate an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; a second capability to integrate electronic images into existing applications without the need to modify the destination application; an interface comprising a software application that enables copying images between physical devices, applications, and the Internet using a single "GO" operation; and a third capability of adding at least one of electronic document and paper processing with a single programming step.

A computer data administration system capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications at least one of locally and via the Internet. The computer data administration system includes at least one memory storing at least one of a common and universal interface protocol for interfacing and communicating; and at least one processor responsively connectable to said at least one memory, and implementing the at least one common and universal interface protocol as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, a computer readable tangible medium stores instructions for implementing a process driven by a computer implemented on at least one of an electronic image, graphics and document administration system capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including at least one of an external device and application at least one of locally and via the Internet. The instructions control the computer to perform the process of: storing at least one of a common and universal interface protocol for interfacing and communicating in at least one memory; and implementing the at least one of common and universal interface protocol as a software application via at least one processor for interfacing and communicating with the plurality of external destinations including the at least one external device and application.

In one embodiment, a computer data administration system includes at least one of an electronic image, graphics and document administration system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The computer data administration system includes a single function copy operation linking devices, applications and the Internet including at least one a go operation, a single function paper copy between devices and software applications, and a single function paper copy between software applications and devices; a one step programming method to add paper support to electronic business processes including at least one of a one step method of supporting paper within electronic business process application optionally including legacy systems with no or minimal reprogramming of the electronic business process application, a method of recreating a module oriented copier in software; and a copier interface implemented as software application including at least one of a virtual copier interface method of presenting to a user an operation of at least one of copying files and electronic images, at least one of to and from, at least one of digital imaging devices and software applications, in a substantially single step, and presenting users with direct access to at least one of tutorial and options from a main application window.

In one embodiment, a server module includes enable virtual copy operation means for initiating, canceling, and resetting said computer data administration system; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data administration system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer data administration system copy operation in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, a computer data administration method includes at least one of an electronic image, graphics and document administration system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The method comprises the steps of integrating an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; integrating electronic images into existing applications without the need to modify the destination application; interfacing via a software application enabling copying images between physical devices, applications, and the Internet using a single "GO" operation; and adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, a server method includes initiating, canceling, and resetting said computer data administration system; maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data administration system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintaining said input, output, and process modules currently being used for a current computer data administration system copy operation in a program object, and saving the currently active modules in a process template file; and maintaining information regarding a current file being copied, and saving the information in a document template file.

A computer information management system includes at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications. The computer information management system is responsively connectable at least one of locally and via the Internet, and includes at least one storage storing a plurality of interface protocols for interfacing and communicating, and at least one processor responsively connectable to the at least one storage. The processor implements the plurality of interface protocols as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, the external devices and applications include, for example, a printer, a facsimile, and a scanner. In one embodiment, the computer information management system includes the capability to integrate an image using software so that the image gets seamlessly replicated and transmitted to at least one of other devices and applications, and via the Internet. In one embodiment, the computer information management system includes the capability to integrate the electronic images into a destination application without the need to modify the destination application.

In one embodiment, the computer information management system includes an interface that enables copying images between physical devices, applications, and the Internet using a single "GO" operation. In one embodiment, the computer information management system includes the capability of adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, the software application includes at least one input module managing information comprising at least one of paper and electronic paper input to the computer information management system, and managing at least one imaging device to input the information through at least one of a scanner and a digital copier, and managing the electronic paper from at least one third-party software applications; at least one output module managing the information output from the computer information management system, managing at least one imaging device to output the information to at least one of a standard Windows printer, an image printer, and a digital copier, and managing the output of the information to the third-party software application; at least one process module applying at least one information processing to the information comprising the at least one of the paper and the electronic paper as it is being copied, applying additional functionality including at least one of workflow and processing functionality to the information comprising the at least one of paper and electronic paper as it is being copied, and applying multiple processes to a single virtual copy; at least one client module presenting the information comprising the at least one of paper and electronic paper as it is being copied, and information related to at least one of the input and output functions; and at least one server module communicable with said at least one input, output, client, and process modules and external applications, and capable of dynamically combining the external applications with at least one of digital capturing devices and digital imaging devices.

In one embodiment, one or more of the external devices and applications integrates the computer information management system into an external application via one of running the computer information management system, as an external service and embedding the computer information management system as an embedded service.

In one embodiment, the server module includes enable virtual copy operation means for initiating, canceling, and resetting said computer information management system; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer information management system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer information management system copy operation in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one mebodiment, the server module includes at least one server module application programmer interface (API). IN one mebodiment, the server module application programmer interface (API) comprises the COM-based interfaces: at least one modules object maintaining a first list of available input, output, and process modules; at least one program object maintaining a second list of currently selected input, output, and process modules; at least one document object maintaining information regarding a current document being copied; at least one system management method object used to initiate, cancel, and reset said computer information management system; and at least one system management event object used to provide feedback to the Client Module.

In one embodiment, a computer information management system includes at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The computer information management system comprises: a first capability to integrate an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; a second capability to integrate electronic images into existing applications without the need to modify the destination application; an interface comprising a software application that enables copying images between physical devices, applications, and the Internet using a single "GO" operation; and a third capability of adding at least one of electronic document and paper processing with a single programming step.

A computer information management system capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications at least one of locally and via the Internet. The computer information management system includes at least one storage storing at least one of a common and universal interface protocol for interfacing and communicating; and at least one processor responsively connectable to said at least one storage, and implementing the at least one common and universal interface protocol as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, a computer readable tangible medium stores instructions for implementing a process driven by a computer implemented on at least one of an electronic image, graphics and document management system capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including at least one of an external device and application at least one of locally and via the Internet. The instructions control the computer to perform the process of: storing at least one of a common and universal interface protocol for interfacing and communicating in at least one storage; and implementing the at least one of common and universal interface protocol as a software application via at least one processor for interfacing and communicating with the plurality of external destinations including the at least one external device and application.

In one embodiment, a computer information management system includes at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The computer information management system includes a single function copy operation linking devices, applications and the Internet including at least one a go operation, a single function paper copy between devices and software applications, and a single function paper copy between software applications and devices; a one step programming method to add paper support to electronic business processes including at least one of a one step method of supporting paper within electronic business process application optionally including legacy systems with no or minimal reprogramming of the electronic business process application, a method of recreating a module oriented copier in software; and a copier interface implemented as software application including at least one of a virtual copier interface method of presenting to a user an operation of at least one of copying files and electronic images, at least one of to and from, at least one of digital imaging devices and software applications, in a substantially single step, and presenting users with direct access to at least one of tutorial and options from a main application window.

In one embodiment, a server module includes enable virtual copy operation means for initiating, canceling, and resetting said computer information management system; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer information management system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer information management system copy operation in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, a computer information management method includes at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The method comprises the steps of integrating an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; integrating electronic images into existing applications without the need to modify the destination application; interfacing via a software application enabling copying images between physical devices, applications, and the Internet using a single "GO" operation; and adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, a server method includes initiating, canceling, and resetting said computer information management system; maintaining a registry containing a list of said input, output, and process modules that can be used in said computer information management system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintaining said input, output, and process modules currently being used for a current computer information management system copy operation in a program object, and saving the currently active modules in a process template file; and maintaining information regarding a current file being copied, and saving the information in a document template file.

A computer data management system includes at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications. The computer data management system is responsively connectable at least one of locally and via the Internet, and includes at least one memory storing a plurality of interface protocols for interfacing and communicating, and at least one processor responsively connectable to the at least one memory. The processor implements at least one interface protocol as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, the external devices and applications include, for example, a printer, a facsimile, and a scanner. In one embodiment, the computer data management system includes the capability to integrate an image using software so that the image gets seamlessly replicated and transmitted to at least one of other devices and applications, and via the Internet. In one embodiment, the computer data management system includes the capability to integrate the electronic images into a destination application without the need to modify the destination application.

In one embodiment, the computer data management system includes an optional interface that enables copying images between physical devices, applications, and the Internet using a single "GO" operation. In one embodiment, the computer data management system includes the optional capability of adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, the software application includes one or more of: at least one input module managing data comprising at least one of paper and electronic paper input to the computer data management system, and managing at least one imaging device to input the data through at least one of a scanner and a digital copier, and managing the electronic paper from at least one third-party software applications; at least one output module managing the data output from the computer data management system, managing at least one imaging device to output the data to at least one of a standard Windows printer, an image printer, and a digital copier, and managing the output of the data to the third-party software application; at least one process module applying at least one data processing to the data comprising the at least one of the paper and the electronic paper as it is being copied, applying additional functionality including at least one of workflow and processing functionality to the data comprising the at least one of paper and electronic paper as it is being copied, and applying multiple processes to a single virtual copy; at least one client module presenting the data comprising the at least one of paper and electronic paper as it is being copied, and information related to at least one of the input and output functions; and at least one server module communicable with said at least one input, output, client, and process modules and external applications, and capable of dynamically combining the external applications with at least one of digital capturing devices and digital imaging devices.

In one embodiment, one or more of the external devices and applications integrates the computer data management system into an external application via at least one of running the computer data management system, as an external service and embedding the computer data management system as an embedded service.

In one embodiment, the server module includes one or more of: enable virtual copy operation means for initiating, canceling, and resetting said computer data management system; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer data management system copy operation in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, the server module includes at least one server module application programmer interface (API). In one embodiment, the server module application programmer interface (API) comprises one or more of the following COM-based interfaces: at least one modules object maintaining a first list of available input, output, and process modules; at least one program object maintaining a second list of currently selected input, output, and process modules; at least one document object maintaining information regarding a current document being copied; at least one system management method object used to initiate, cancel, and reset said computer data management system; and at least one system management event object used to provide feedback to the Client Module.

In one embodiment, a computer data management system includes at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The computer data management system comprises one or more of: a first capability to integrate an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; a second capability to integrate electronic images into existing applications without the need to modify the destination application; an interface comprising a software application that enables copying images between physical devices, applications, and the Internet using a single "GO" operation; and a third capability of adding at least one of electronic document and paper processing with a single programming step.

A computer data management system capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications at least one of locally and via the Internet. The computer data management system includes at least one memory storing at least one of a common and universal interface protocol for interfacing and communicating; and at least one data processor responsively connectable to said at least one memory, and implementing the at least one common and universal interface protocol as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, a computer readable tangible medium stores instructions for implementing a process driven by a computer implemented on at least one of an electronic image, graphics and document management system capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including at least one of an external device and application at least one of locally and via the Internet. The instructions control the computer to perform the process of: storing at least one of a common and universal interface protocol for interfacing and communicating in at least one memory; and implementing the at least one of common and universal interface protocol interfacing and communicating with the plurality of external destinations.

In one embodiment, a computer data management system includes at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The computer data management system includes one or more of: a single function copy operation linking devices, applications and the Internet including at least one a go operation, a single function paper copy between devices and software applications, and a single function paper copy between software applications and devices; a one step programming method to add paper support to electronic business processes including at least one of a one step method of supporting paper within electronic business process application optionally including legacy systems with no or minimal reprogramming of the electronic business process application, a method of recreating a module oriented copier in software; and a copier interface implemented as software application including at least one of a virtual copier interface method of presenting to a user an operation of at least one of copying files and electronic images, at least one of to and from, at least one of digital imaging devices and software applications, in a substantially single step, and presenting users with direct access to at least one of tutorial and options from a main application window.

In one embodiment, a server module includes one or more of: enable virtual copy operation means for initiating, canceling, and resetting said computer data management system; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer data management system copy operation in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, a computer data management method includes at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The method comprises one or more of the steps of integrating an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; integrating electronic images into existing applications without the need to modify the destination application; interfacing via a software application enabling copying images between physical devices, applications, and the Internet using a single "GO" operation; and adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, a server method includes one or more of: initiating, canceling, and resetting said computer data management system; maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintaining said input, output, and process modules currently being used for a current computer data management system copy operation in a program object, and saving the currently active modules in a process template file; and maintaining information regarding a current file being copied, and saving the information in a document template file.

A computer data administration system includes at least one of an electronic image, graphics and document administration system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications. The computer

data administration system is responsively connectable at least one of locally and via the Internet, and includes at least one memory storing a plurality of interface protocols for interfacing and communicating, and at least one processor responsively connectable to the at least one memory. The processor implements at least one interface protocol as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, the external devices and applications include, for example, a printer, a facsimile, and a scanner. In one embodiment, the computer data administration system includes the capability to integrate an image using software so that the image gets seamlessly replicated and transmitted to at least one of other devices and applications, and via the Internet. In one embodiment, the computer data administration system includes the capability to integrate the electronic images into a destination application without the need to modify the destination application.

In one embodiment, the computer data administration system includes an optional interface that enables copying images between physical devices, applications, and the Internet using a single "GO" operation. In one embodiment, the computer data administration system includes the optional capability of adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, the software application includes one or more of: at least one input module managing data comprising at least one of paper and electronic paper input to the computer data administration system, and managing at least one imaging device to input the data through at least one of a scanner and a digital copier, and managing the electronic paper from at least one third-party software applications; at least one output module managing the data output from the computer data administration system, managing at least one imaging device to output the data to at least one of a standard Windows printer, an image printer, and a digital copier, and managing the output of the data to the third-party software application; at least one process module applying at least one data processing to the data comprising the at least one of the paper and the electronic paper as it is being copied, applying additional functionality including at least one of workflow and processing functionality to the data comprising the at least one of paper and electronic paper as it is being copied, and applying multiple processes to a single virtual copy; at least one client module presenting the data comprising the at least one of paper and electronic paper as it is being copied, and information related to at least one of the input and output functions; and at least one server module communicable with said at least one input, output, client, and process modules and external applications, and capable of dynamically combining the external applications with at least one of digital capturing devices and digital imaging devices.

In one embodiment, one or more of the external devices and applications integrates the computer data administration system into an external application via at least one of running the computer data administration system, as an external service and embedding the computer data administration system as an embedded service.

In one embodiment, the server module includes one or more of: enable virtual copy operation means for initiating, canceling, and resetting said computer data administration system; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data administration system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer data administration system copy operation in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, the server module includes at least one server module application programmer interface (API). In one embodiment, the server module application programmer interface (API) comprises one or more of the following COM-based interfaces: at least one modules object maintaining a first list of available input, output, and process modules; at least one program object maintaining a second list of currently selected input, output, and process modules; at least one document object maintaining information regarding a current document being copied; at least one system administration method object used to initiate, cancel, and reset said computer data administration system; and at least one system administration event object used to provide feedback to the Client Module.

In one embodiment, a computer data administration system includes at least one of an electronic image, graphics and document administration system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The computer data administration system comprises one or more of: a first capability to integrate an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; a second capability to integrate electronic images into existing applications without the need to modify the destination application; an interface comprising a software application that enables copying images between physical devices, applications, and the Internet using a single "GO" operation; and a third capability of adding at least one of electronic document and paper processing with a single programming step.

A computer data administration system capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications at least one of locally and via the Internet. The computer data administration system includes at least one memory storing at least one of a common and universal interface protocol for interfacing and communicating; and at least one data processor responsively connectable to said at least one memory, and implementing the at least one common and universal interface protocol as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, a computer readable tangible medium stores instructions for implementing a process driven by a computer implemented on at least one of an electronic image, graphics and document administration system capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including at least one of an external device and application at least one of locally and via the Internet. The instructions control the computer to perform the process of: storing at least one of a common and universal interface protocol for interfacing and communicating in at least one memory; and implementing the at least one of common and universal interface protocol interfacing and communicating with the plurality of external destinations.

In one embodiment, a computer data administration system includes at least one of an electronic image, graphics and document administration system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The computer data administration system includes one or more of: a single function copy operation linking devices, applications and the Internet including at least one a go operation, a single function paper copy between devices and software applications, and a single function paper copy between software applications and devices; a one step programming method to add paper support to electronic business processes including at least one of a one step method of supporting paper within electronic business process application optionally including legacy systems with no or minimal reprogramming of the electronic business process application, a method of recreating a module oriented copier in software; and a copier interface implemented as software application including at least one of a virtual copier interface method of presenting to a user an operation of at least one of copying files and electronic images, at least one of to and from, at least one of digital imaging devices and software applications, in a substantially single step, and presenting users with direct access to at least one of tutorial and options from a main application window.

In one embodiment, a server module includes one or more of: enable virtual copy operation means for initiating, canceling, and resetting said computer data administration system; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data administration system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer data administration system copy operation in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, a computer data administration method includes at least one of an electronic image, graphics and document administration system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The method comprises one or more of the steps of integrating an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; integrating electronic images into existing applications without the need to modify the destination application; interfacing via a software application enabling copying images between physical devices, applications, and the Internet using a single "GO" operation; and adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, a server method includes one or more of: initiating, canceling, and resetting said computer data administration system; maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data administration system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintaining said input, output, and process modules currently being used for a current computer data administration system copy operation in a program object, and saving the currently active modules in a process template file; and maintaining information regarding a current file being copied, and saving the information in a document template file.

A workstation data management system includes at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications. The workstation data management system is responsively connectable at least one of locally and via the Internet, and includes at least one memory storing a plurality of interface protocols for interfacing and communicating, and at least one processor responsively connectable to the at least one memory. The processor implements at least one interface protocol as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, the external devices and applications include, for example, a printer, a facsimile, and a scanner. In one embodiment, the workstation data management system includes the capability to integrate an image using software so that the image gets seamlessly replicated and transmitted to at least one of other devices and applications, and via the Internet. In one embodiment, the workstation data management system includes the capability to integrate the electronic images into a destination application without the need to modify the destination application.

In one embodiment, the workstation data management system includes an optional interface that enables copying images between physical devices, applications, and the Internet using a single "GO" operation. In one embodiment, the workstation data management system includes the optional capability of adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, the software application includes one or more of: at least one input module managing data comprising at least one of paper and electronic paper input to the workstation data management system, and managing at least one imaging device to input the data through at least one of a scanner and a digital copier, and managing the electronic paper from at least one third-party software applications; at least one output module managing the data output from the workstation data management system, managing at least one imaging device to output the data to at least one of a standard Windows printer, an image printer, and a digital copier, and managing the output of the data to the third-party software application; at least one process module applying at least one data processing to the data comprising the at least one of the paper and the electronic paper as it is being copied, applying additional functionality including at least one of workflow and processing functionality to the data comprising the at least one of paper and electronic paper as it is being copied, and applying multiple processes to a single virtual copy; at least one client module

presenting the data comprising the at least one of paper and electronic paper as it is being copied, and information related to at least one of the input and output functions; and at least one server module communicable with said at least one input, output, client, and process modules and external applications, and capable of dynamically combining the external applications with at least one of digital capturing devices and digital imaging devices.

In one embodiment, one or more of the external devices and applications integrates the workstation data management system into an external application via at least one of running the workstation data management system, as an external service and embedding the workstation data management system as an embedded service.

In one embodiment, the server module includes one or more of: enable virtual copy operation means for initiating, canceling, and resetting said workstation data management system; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said workstation data management system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current workstation data management system copy operation in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, the server module includes at least one server module application programmer interface (API). In one embodiment, the server module application programmer interface (API) comprises one or more of the following COM-based interfaces: at least one modules object maintaining a first list of available input, output, and process modules; at least one program object maintaining a second list of currently selected input, output, and process modules; at least one document object maintaining information regarding a current document being copied; at least one system management method object used to initiate, cancel, and reset said workstation data management system; and at least one system management event object used to provide feedback to the Client Module.

In one embodiment, a workstation data management system includes at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The workstation data management system comprises one or more of: a first capability to integrate an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; a second capability to integrate electronic images into existing applications without the need to modify the destination application; an interface comprising a software application that enables copying images between physical devices, applications, and the Internet using a single "GO" operation; and a third capability of adding at least one of electronic document and paper processing with a single programming step.

A workstation data management system capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications at least one of locally and via the Internet. The workstation data management system includes at least one memory storing at least one of a common and universal interface protocol for interfacing and communicating; and at least one data processor responsively connectable to said at least one memory, and implementing the at least one common and universal interface protocol as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, a workstation readable tangible medium stores instructions for implementing a process driven by a workstation implemented on at least one of an electronic image, graphics and document management system capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including at least one of an external device and application at least one of locally and via the Internet. The instructions control the workstation to perform the process of: storing at least one of a common and universal interface protocol for interfacing and communicating in at least one memory; and implementing the at least one of common and universal interface protocol interfacing and communicating with the plurality of external destinations.

In one embodiment, a workstation data management system includes at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The workstation data management system includes one or more of: a single function copy operation linking devices, applications and the Internet including at least one a go operation, a single function paper copy between devices and software applications, and a single function paper copy between software applications and devices; a one step programming method to add paper support to electronic business processes including at least one of a one step method of supporting paper within electronic business process application optionally including legacy systems with no or minimal reprogramming of the electronic business process application, a method of recreating a module oriented copier in software; and a copier interface implemented as software application including at least one of a virtual copier interface method of presenting to a user an operation of at least one of copying files and electronic images, at least one of to and from, at least one of digital imaging devices and software applications, in a substantially single step, and presenting users with direct access to at least one of tutorial and options from a main application window.

In one embodiment, a server module includes one or more of: enable virtual copy operation means for initiating, canceling, and resetting said workstation data management system; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said workstation data management system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current workstation data management system copy operation in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, a workstation data management method includes at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The method comprises one or more of the steps of integrating an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; integrating electronic images into existing applications without the need to modify the destination application; interfacing via a software application enabling copying images between physical devices, applications, and the Internet using a single "GO" operation; and adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, a server method includes one or more of: initiating, canceling, and resetting said workstation data management system; maintaining a registry containing a list of said input, output, and process modules that can be used in said workstation data management system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintaining said input, output, and process modules currently being used for a current workstation data management system copy operation in a program object, and saving the currently active modules in a process template file; and maintaining information regarding a current file being copied, and saving the information in a document template file.

A computer data management apparatus includes at least one of an electronic image, graphics and document management apparatus capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications. The computer data management apparatus is responsively connectable at least one of locally and via the Internet, and includes at least one memory storing a plurality of interface protocols for interfacing and communicating, and at least one processor responsively connectable to the at least one memory. The processor implements at least one interface protocol as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, the external devices and applications include, for example, a printer, a facsimile, and a scanner. In one embodiment, the computer data management apparatus includes the capability to integrate an image using software so that the image gets seamlessly replicated and transmitted to at least one of other devices and applications, and via the Internet. In one embodiment, the computer data management apparatus includes the capability to integrate the electronic images into a destination application without the need to modify the destination application.

In one embodiment, the computer data management apparatus includes an optional interface that enables copying images between physical devices, applications, and the Internet using a single "GO" operation. In one embodiment, the computer data management apparatus includes the optional capability of adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, the software application includes one or more of: at least one input module managing data comprising at least one of paper and electronic paper input to the computer data management apparatus, and managing at least one imaging device to input the data through at least one of a scanner and a digital copier, and managing the electronic paper from at least one third-party software applications; at least one output module managing the data output from the computer data management apparatus, managing at least one imaging device to output the data to at least one of a standard Windows printer, an image printer, and a digital copier, and managing the output of the data to the third-party software application; at least one process module applying at least one data processing to the data comprising the at least one of the paper and the electronic paper as it is being copied, applying additional functionality including at least one of workflow and processing functionality to the data comprising the at least one of paper and electronic paper as it is being copied, and applying multiple processes to a single virtual copy; at least one client module presenting the data comprising the at least one of paper and electronic paper as it is being copied, and information related to at least one of the input and output functions; and at least one server module communicable with said at least one input, output, client, and process modules and external applications, and capable of dyuamically combining the external applications with at least one of digital capturing devices and digital imaging devices.

In one embodiment, one or more of the external devices and applications integrates the computer data management apparatus into an external application via at least one of running the computer data management apparatus, as an external service and embedding the computer data management apparatus as an embedded service.

In one embodiment, the server module includes one or more of: enable virtual copy operation means for initiating, canceling, and resetting said computer data management apparatus; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management apparatus, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer data management apparatus copy operation in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, the server module includes at least one server module application programmer interface (API). In one embodiment, the server module application programmer interface (API) comprises one or more of the following COM-based interfaces: at least one modules object maintaining a first list of available input, output, and process modules; at least one program object maintaining a second list of currently selected input, output, and process modules; at least one document object maintaining information regarding a current document being copied; at least one apparatus management method object used to initiate, cancel, and reset said computer data management apparatus; and at least one apparatus management event object used to provide feedback to the Client Module.

In one embodiment, a computer data management apparatus includes at least one of an electronic image, graphics

and document management apparatus capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The computer data management apparatus comprises one or more of: a first capability to integrate an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; a second capability to integrate electronic images into existing applications without the need to modify the destination application; an interface comprising a software application that enables copying images between physical devices, applications, and the Internet using a single "GO" operation; and a third capability of adding at least one of electronic document and paper processing with a single programming step.

A computer data management apparatus capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications at least one of locally and via the Internet. The computer data management apparatus includes at least one memory storing at least one of a common and universal interface protocol for interfacing and communicating; and at least one data processor responsively connectable to said at least one memory, and implementing the at least one common and universal interface protocol as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, a computer readable tangible medium stores instructions for implementing a process driven by a computer implemented on at least one of an electronic image, graphics and document management apparatus capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including at least one of an external device and application at least one of locally and via the Internet. The instructions control the computer to perform the process of: storing at least one of a common and universal interface protocol for interfacing and communicating in at least one memory; and implementing the at least one of common and universal interface protocol interfacing and communicating with the plurality of external destinations.

In one embodiment, a computer data management apparatus includes at least one of an electronic image, graphics and document management apparatus capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The computer data management apparatus includes one or more of: a single function copy operation linking devices, applications and the Internet including at least one a go operation, a single function paper copy between devices and software applications, and a single function paper copy between software applications and devices; a one step programming method to add paper support to electronic business processes including at least one of a one step method of supporting paper within electronic business process application optionally including legacy apparatus with no or minimal reprogramming of the electronic business process application, a method of recreating a module oriented copier in software; and a copier interface implemented as software application including at least one of a virtual copier interface method of presenting to a user an operation of at least one of copying files and electronic images, at least one of to and from, at least one of digital imaging devices and software applications, in a substantially single step, and presenting users with direct access to at least one of tutorial and options from a main application window.

In one embodiment, a server module includes one or more of: enable virtual copy operation means for initiating, canceling, and resetting said computer data management apparatus; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management apparatus, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer data management apparatus copy operation in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, a computer data management method includes at least one of an electronic image, graphics and document management apparatus capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The method comprises one or more of the steps of integrating an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; integrating electronic images into existing applications without the need to modify the destination application; interfacing via a software application enabling copying images between physical devices, applications, and the Internet using a single "GO" operation; and adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, a server method includes one or more of: initiating, canceling, and resetting said computer data management apparatus; maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management apparatus, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintaining said input, output, and process modules currently being used for a current computer data management apparatus copy operation in a program object, and saving the currently active modules in a process template file; and maintaining information regarding a current file being copied, and saving the information in a document template file.

A computer data management device includes at least one of an electronic image, graphics and document management device capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications. The computer data management device is responsively connectable at least one of locally and via the Internet, and includes at least one memory storing a plurality of interface procedures for communicating and communicating, and at least one processor responsively connectable to the at least one memory. The processor implements at least one interface procedure as

a software application for communicating and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, the external devices and applications include, for example, a printer, a facsimile, and a scanner. In one embodiment, the computer data management device includes the capability to integrate an image using software so that the image gets seamlessly replicated and transmitted to at least one of other devices and applications, and via the Internet. In one embodiment, the computer data management device includes the capability to integrate the electronic images into a destination application without the need to modify the destination application.

In one embodiment, the computer data management device includes an optional interface that enables copying images between physical devices, applications, and the Internet using a single "GO" action. In one embodiment, the computer data management device includes the optional capability of adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, the software application includes one or more of: at least one input module managing data comprising at least one of paper and electronic paper input to the computer data management device, and managing at least one imaging device to input the data through at least one of a scanner and a digital copier, and managing the electronic paper from at least one third-party software applications; at least one output module managing the data output from the computer data management device, managing at least one imaging device to output the data to at least one of a standard Windows printer, an image printer, and a digital copier, and managing the output of the data to the third-party software application; at least one process module applying at least one data processing to the data comprising the at least one of the paper and the electronic paper as it is being copied, applying additional functionality including at least one of workflow and processing functionality to the data comprising the at least one of paper and electronic paper as it is being copied, and applying multiple processes to a single virtual copy; at least one client module presenting the data comprising the at least one of paper and electronic paper as it is being copied, and information related to at least one of the input and output functions; and at least one server module communicable with said at least one input, output, client, and process modules and external applications, and capable of dynamically combining the external applications with at least one of digital capturing devices and digital imaging devices.

In one embodiment, one or more of the external devices and applications integrates the computer data management device into an external application via at least one of running the computer data management device, as an external service and embedding the computer data management device as an embedded service.

In one embodiment, the server module includes one or more of: enable virtual copy action means for initiating, canceling, and resetting said computer data management device; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management device, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer data management device copy action in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, the server module includes at least one server module application programmer interface (API). In one embodiment, the server module application programmer interface (API) comprises one or more of the following COM-based interfaces: at least one modules is object maintaining a first list of available input, output, and process modules; at least one program object maintaining a second list of currently selected input, output, and process modules; at least one document object maintaining information regarding a current document being copied; at least one device management method object used to initiate, cancel, and reset said computer data management device; and at least one device management event object used to provide feedback to the Client Module.

In one embodiment, a computer data management device includes at least one of an electronic image, graphics and document management device capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The computer data management device comprises one or more of: a first capability to integrate an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; a second capability to integrate electronic images into existing applications without the need to modify the destination application; an interface comprising a software application that enables copying images between physical devices, applications, and the Internet using a single "GO" action; and a third capability of adding at least one of electronic document and paper processing with a single programming step.

A computer data management device capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications at least one of locally and via the Internet. The computer data management device includes at least one memory storing at least one of a common and universal interface procedure for communicating and communicating; and at least one data processor responsively connectable to said at least one memory, and implementing the at least one common and universal interface procedure as a software application for communicating and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, a computer readable tangible medium stores instructions for implementing a process driven by a computer implemented on at least one of an electronic image, graphics and document management device capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including at least one of an external device and application at least one of locally and via the Internet. The instructions control the computer to perform the process of: storing at least one of a common and universal interface procedure for communicating and communicating in at least one memory; and implementing the at least one of common and universal interface procedure communicating and communicating with the plurality of external destinations.

In one embodiment, a computer data management device includes at least one of an electronic image, graphics and

document management device capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The computer data management device includes one or more of: a single function copy action linking devices, applications and the Internet including at least one a go action, a single function paper copy between devices and software applications, and a single function paper copy between software applications and devices; a one step programming method to add paper support to electronic business processes including at least one of a one step method of supporting paper within electronic business process application optionally including legacy devices with no or minimal reprogramming of the electronic business process application, a method of recreating a module oriented copier in software; and a copier interface implemented as software application including at least one of a virtual copier interface method of presenting to a user an action of at least one of copying files and electronic images, at least one of to and from, at least one of digital imaging devices and software applications, in a substantially single step, and presenting users with direct access to at least one of tutorial and options from a main application window.

In one embodiment, a server module includes one or more of: enable virtual copy action means for initiating, canceling, and resetting said computer data management device; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management device, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer data management device copy action in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, a computer data management method includes at least one of an electronic image, graphics and document management device capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The method comprises one or more of the steps of integrating an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; integrating electronic images into existing applications without the need to modify the destination application; communicating via a software application enabling copying images between physical devices, applications, and the Internet using a single "GO" action; and adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, a server method includes one or more of: initiating, canceling, and resetting said computer data management device; maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management device, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintaining said input, output, and process modules currently being used for a current computer data management device copy action in a program object, and saving the currently active modules in a process template file; and maintaining information regarding a current file being copied, and saving the information in a document template file.

A computer data management method includes at least one of an electronic image, graphics and document management method capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications. The computer data management method is connectable at least one of locally and via the Internet, and accesses at least one memory storing a plurality of interface protocols for interfacing and communicating, and at least one processor responsively connectable to the at least one memory. The processor implements at least one interface protocol as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, the external devices and applications include, for example, a printer, a facsimile, and a scanner. In one embodiment, the computer data management method includes the capability to integrate an image using software so that the image gets seamlessly replicated and transmitted to at least one of other devices and applications, and via the Internet. In one embodiment, the computer data management method includes the capability to integrate the electronic images into a destination application without the need to modify the destination application.

In one embodiment, the computer data management method includes an optional interface that enables copying images between physical devices, applications, and the Internet using a single "GO" operation. In one embodiment, the computer data management method includes the optional capability of adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, the software application includes one or more of: managing data comprising at least one of paper and electronic paper input to the computer data management method, and managing at least one imaging device to input the data through at least one of a scanner and a digital copier, and managing the electronic paper from at least one third-party software applications; managing the data output from the computer data management method, managing at least one imaging device to output the data to at least one of a standard Windows printer, an image printer, and a digital copier, and managing the output of the data to the third-party software application; applying at least one data processing to the data comprising the at least one of the paper and the electronic paper as it is being copied, applying additional functionality including at least one of workflow and processing functionality to the data comprising the at least one of paper and electronic paper as it is being copied, and applying multiple processes to a single virtual copy; presenting the data comprising the at least one of paper and electronic paper as it is being copied, and information related to at least one of the input and output functions; and communicable with said at least one input, output, client, and process modules and external applications, and capable of dynamically combining the external applications with at least one of digital capturing devices and digital imaging devices.

In one embodiment, one or more of the external devices and applications integrates the computer data management method into an external application via at least one of running the computer data management method, as an external service and embedding the computer data management method as an embedded service.

In one embodiment, the server module includes one or more of: enable virtual copy operation means for initiating, canceling, and resetting said computer data management method; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management method, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer data management method copy operation in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, the server module includes at least one server module application programmer interface (API). In one embodiment, the server module application programmer interface (API) comprises one or more of the following COM-based interfaces: at least one modules object maintaining a first list of available input, output, and process modules; at least one program object maintaining a second list of currently selected input, output, and process modules; at least one document object maintaining information regarding a current document being copied; at least one method management method object used to initiate, cancel, and reset said computer data management method; and at least one method management event object used to provide feedback to the Client Module.

In one embodiment, a computer data management method includes at least one of an electronic image, graphics and document management method capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The computer data management method comprises one or more of: a first capability to integrate an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the Internet; a second capability to integrate electronic images into existing applications without the need to modify the destination application; an interface comprising a software application that enables copying images between physical devices, applications, and the Internet using a single "GO" operation; and a third capability of adding at least one of electronic document and paper processing with a single programming step.

A computer data management method capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications at least one of locally and via the Internet. The computer data management method includes at least one memory storing at least one of a common and universal interface protocol for interfacing and communicating; and at least one data processor responsively connectable to said at least one memory, and implementing the at least one common and universal interface protocol as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications.

In one embodiment, a computer readable tangible medium stores instructions for implementing a process driven by a computer implemented on at least one of an electronic image, graphics and document management method capable of managing and transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including at least one of an external device and application at least one of locally and via the Internet. The instructions control the computer to perform the process of: storing at least one of a common and universal interface protocol for interfacing and communicating in at least one memory; and implementing the at least one of common and universal interface protocol interfacing and communicating with the plurality of external destinations.

In one embodiment, a computer data management method includes at least one of an electronic image, graphics and document management method capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The computer data management method includes one or more of: a single function copy operation linking devices, applications and the Internet including at least one a go operation, a single function paper copy between devices and software applications, and a single function paper copy between software applications and devices; a one step programming method to add paper support to electronic business processes including at least one of a one step method of supporting paper within electronic business process application optionally including legacy methods with no or minimal reprogramming of the electronic business process application, a method of recreating a module oriented copier in software; and a copier interface implemented as software application including at least one of a virtual copier interface method of presenting to a user an operation of at least one of copying files and electronic images, at least one of to and from, at least one of digital imaging devices and software applications, in a substantially single step, and presenting users with direct access to at least one of tutorial and options from a main application window.

In one embodiment, a server module includes one or more of: enable virtual copy operation means for initiating, canceling, and resetting said computer data management method; maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management method, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer data management method copy operation in a program object, and saving the currently active modules in a process template file; and maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

In one embodiment, a computer data management method includes at least one of an electronic image, graphics and document management method capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet. The method comprises one or more of the steps of integrating an image using software so that the image gets seamlessly replicated into at least one of other devices and applications, and via the

Internet; integrating electronic images into existing applications without the need to modify the destination application; interfacing via a software application enabling copying images between physical devices, applications, and the Internet using a single "GO" operation; and adding at least one of electronic document and paper processing with a single programming step.

In one embodiment, a server method includes one or more of: initiating, canceling, and resetting said computer data management method; maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management method, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules; maintaining said input, output, and process modules currently being used for a current computer data management method copy operation in a program object, and saving the currently active modules in a process template file; and maintaining information regarding a current file being copied, and saving the information in a document template file.

In accordance with another embodiment of the invention, a computer readable tangible medium is provided that stores an object thereon, for execution by the computer.

There has thus been outlined, rather broadly, the more important features of the invention in order that the detailed description thereof that follows may be better understood, and in order that the present contribution to the art may be better appreciated. There are, of course, additional features of the invention that will be described hereinafter and which will form the subject matter of the claims appended hereto. In this respect, before explaining at least one embodiment of the invention in detail, it is to be understood that the invention is not limited in its application to the details of construction and to the arrangements of the components set forth in the following description or illustrated in the drawings. The invention is capable of other embodiments and of being practiced and carried out in various ways. Also, it is to be understood that the phraseology and terminology employed herein are for the purpose of description and should not be regarded as limiting.

As such, those skilled in the art will appreciate that the conception, upon which this disclosure is based, may readily be utilized as a basis for the designing of other structures, methods and systems for carrying out the several purposes of the present invention. It is important, therefore, that the claims be regarded as including such equivalent constructions insofar as they do not depart from the spirit and scope of the present invention.

Further, the purpose of the foregoing abstract is to enable the U.S. Patent and Trademark Office and the public generally, and especially the scientists, engineers and practitioners in the art who are not familiar with patent or legal terms or phraseology, to determine quickly from a cursory inspection the nature and essence of the technical disclosure of the application. The abstract is neither intended to define the invention of the application, which is measured by the claims, nor is it intended to be limiting as to the scope of the invention in any way.

These together with other objects of the invention, along with the various features of novelty which characterize the invention, are pointed out with particularity in the claims annexed to and forming a part of this disclosure. For a better understanding of the invention, its operating advantages and the specific objects attained by its uses, reference should be made to the accompanying drawings and descriptive matter in which there is illustrated preferred embodiments of the invention.

These together with other objects and advantages which will be subsequently apparent, reside in the details of construction and operation as more fully herein described and claimed, with reference being had to the accompanying drawings forming a part hereof wherein like numerals refer to like elements throughout.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is an illustration of the placement and/or use of the computer architecture and/or method of the present invention;

FIG. **2** is an illustration of the component factory migrating the original "C"-level API from its original state into the generic interface defined by the topmost layer;

FIG.**3** is an overview of the computer architecture in the present invention;

FIG. **4** is an illustration of the design of an Object in accordance with the computer architecture of the present invention;

FIG. **5** is an illustration of the architecture comprised of two major parts;

FIG. **6** is an illustration of the architecture of an engine component including, for example, three layers designed to migrate the original API of the engine to a consistent COM interface;

FIG. **7** is a table illustrating the engine management specification with definitions;

FIG. **8** is an illustration of the engine management layer being divided into three functions/specifications;

FIG. **9** is an illustration of exemplary tables used to drive the three functions of the engine management layer illustrated in FIG. **8**;

FIG. **10** is an exemplary table illustrating the engine configuration specification with definitions;

FIG. **11** is another exemplary table illustrating the engine configuration specification;

FIG. **12** is an exemplary table illustrating the engine functionality specification with definitions;

FIG. **13** is another exemplary table illustrating the engine functionality specification;

FIG. **14** is an illustration of a main central processing unit for implementing the computer processing in accordance with a computer implemented embodiment of the present invention;

FIG. **15** illustrates a block diagram of the internal hardware of the computer of FIG. **14**;

FIG. **16** is a block diagram of the internal hardware of the computer of FIG. **15** in accordance with a second embodiment;

FIG. **17** is an illustration of an exemplary memory medium which can be used with disk drives illustrated in FIGS. **14**–**16**;

FIG. **18** is an illustration of another embodiment of the component factory migrating the original "C"-level API from its original state into the generic interface defined by the topmost layer;

FIG. **19** is an illustration of a distributed environment or architecture for manually and/or automatically generating and/or using reusable software components for client server and/or intranet operating environments;

FIG. **20** is a detailed illustration of the distributed environment or architecture for manually and/or automatically generating and/or using reusable software components for client server and/or intranet operating environments;

FIG. **21** is an illustration of a distributed environment or architecture for manually and/or automatically generating and/or using reusable software components for network environments, such as the Internet;

FIG. **22** is a detailed illustration of the distributed environment or architecture for manually and/or automatically generating and/or using reusable software components in the Internet environment;

FIGS. **23**A–**23**C are illustrations of the image viewer user interface and/or functionality associated therewith in accordance with the present invention;

FIG. **24** is an illustration of a stand-alone and/or distributed environment or architecture for image viewer in client server and/or intranet operating environments;

FIG. **25** is a detailed illustration of a stand-alone and/or distributed environment or architecture for image viewer in client server and/or intranet operating environments;

FIG. **26** is an illustration of a stand-alone and/or distributed environment or architecture for image viewer in network environments, such as the Internet;

FIG. **27** is a detailed illustration of a stand-alone and/or distributed environment or architecture for image viewer in the Internet environment;

FIGS. **28** and **29** are illustrations of the interface of the Virtual Copier (VC) embodiment of the present invention with a Go button much like a physical copier;

FIG. **30** is an illustration of the sequence used with Virtual Copier with just the Power VC portion of the main Virtual Copier window;

FIG. **31** is an illustration of the five core modules of VC;

FIG. **32** is an illustration of VC recognizing that the third-party application is running, and intelligently copying paper to and from that application;

FIG. **33** is an illustration of a button that can be placed on a third-party application that launches VC in the background;

FIG. **34** is an illustration of the VC logic flow;

FIG. **35** is an illustration of VC updating its Client Module as well as the results of each Module acting on the document;

FIG. **36** is an illustration of the structure of the Modules Object;

FIG. **37** is an illustration of the structure of the Program Object;

FIG. **38** is an illustration of the internal VDocument mapping to physical files;

FIG. **39** is an illustration of the VDocument Object;

FIGS. **40** and **41** are illustrations of two events that the Server Module supports: Error and Status, the Error event being generated anytime any of the Modules produce an error condition, and the Status event being generated when information needs to be transferred between the IOP or Server Modules and the Client Module;

FIG. **42** is an illustration of a general workflow of the events that are generated that manage the flow of modules and user interaction with the Server Module;

FIG. **43** is an illustration of the general logic flow of the Client Module;

FIG. **44** is an illustration of the basic Client architecture;

FIG. **45** is an illustration of the API for the Input, Process, and Output Modules that are made simple so that third-party vendors can create their own custom versions of these modules with relative ease;

FIGS. **46**–**47** are illustrations of the Feedback object used to communicate between the IOP and the Server Module; and

FIG. **48** is an illustration of the basic IOP architecture.

NOTATIONS AND NOMENCLATURE

The detailed descriptions which follow may be presented in terms of program procedures executed on a computer or network of computers. These procedural descriptions and representations are the means used by those skilled in the art to most effectively convey the substance of their work to others skilled in the art.

A procedure is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. These steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared and otherwise manipulated. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. It should be noted, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

Further, the manipulations performed are often referred to in terms, such as adding or comparing, which are commonly associated with mental operations performed by a human operator. No such capability of a human operator is necessary, or desirable in most cases, in any of the operations described herein which form part of the present invention; the operations are machine operations. Of course, one or more of the above operations may alternatively be done manually. Useful machines for performing the operation of the present invention include general purpose digital computers or similar devices.

The present invention also relates to apparatus for performing these operations. This apparatus may be specially constructed for the required purpose or it may comprise a general purpose computer as selectively activated or reconfigured by a computer program stored in the computer. The procedures presented herein are not inherently related to a particular computer or other apparatus. Various general purpose machines may be used with programs written in accordance with the teachings herein, or it may prove more convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these machines will appear from the description given.

BEST MODE FOR CARRYING OUT THE INVENTION

Reference now will be made in detail to the presently preferred embodiments of the invention. Such embodiments are provided by way of explanation of the invention, which is not intended to be limited thereto.

In fact, those of ordinary skill in the art may appreciate upon reading the present specification and viewing the present drawings that various modifications and variations can be made. For example, features illustrated or described as part of one embodiment can be used on other embodiments to yield a still further embodiment. Additionally, certain features may be interchanged with similar devices or features not mentioned yet which perform the same or similar functions. It is therefore intended that such modifications and variations are included within the totality of the present invention.

The purpose of the Virtual Copier ("VC") aspect of the present invention is to enable a typical PC user to add electronic paper processing to their existing business process. VC is an extension of the concept we understand as copying. In its simplest form it extends the notion of copying from a process that involves paper going through a conventional copier device, to a process that involves paper being scanned from a device at one location and copied to a device at another location. In its more sophisticated form, VC can copy paper from a device at one location directly into a business application residing on a network or on the Internet, or visa versa. The VC invention is software that manages paper so that it can be electronically and seamlessly copied in and out of devices and business applications (such as Microsoft Office, Microsoft Exchange, Lotus Notes) with an optional single-step Go operation. The VC software can reside on a PC, LAN/WAN server, digital device (such as a digital copier), or on a web server to be accessed over the Internet.

Virtual Copier is designed to solve the corporate paper problem by enabling existing web-based and client-server applications to manage paper as part of their solution. Virtual Copier links the familiar and universal world of paper and digital devices to web-based and client-server applications. The result is that the automated business processes become the primary storage of paper in electronic form. Information that is typically managed and processed in paper form is "copied" into the system and managed by the business processes with which users are accustomed, which is made possible by using Virtual Copier. Simple extensions of Virtual Copier support seamless electronic outsourcing of paper processing and archival services over the web.

Virtual Copier is a unique combination of an intuitive application built on an open component architecture that delivers a simple innovation: provide paper processing to existing Intranet and client-server business processes without any fuss. Whether it is an office clerk that needs to easily copy a report from a desktop scanner to the company's Intranet-networked copier, or an accounting software integrator that wants to embed paper processing, Virtual Copier offers a simple solution. To the office clerk Virtual Copier is a document imaging application packaged in the familiar setting of an office copier. To the integrator, the underlying open architecture of Virtual Copier offers a simple integration path for embedding paper processing into its client-server or web-based software solution.

Although managing paper manually is one of the great problems facing corporations, there has been little innovation in enabling those workers to eliminate the need to continuously work with paper manually. Much of the problem stems from the complexity of traditional document management systems, which require days of training and months to become familiar with the system in order to be proficient. Virtual Copier was designed to be as simple as a copier to operate, and yet still provide the complete capability of integrating paper with existing business applications. By simplifying the interface and underlying software infrastructure, VC can manage paper in electronic form as easily as is currently done in physical form.

VC extends the notion of a copier, which simply replicates the image of an original document onto another piece of paper using a single GO or START button, to do a similar operation in software so that the image gets seamlessly replicated into other devices or applications or the Internet.

An example of this is the actual implementation of Virtual Copier as a consumer product. The interface of the consumer product called Virtual Copier has a Go button much like a physical copier. This GO button can copy paper, whether physical or electronic, from one device and or application to another device and/or application.

What makes Virtual Copier as simple as its physical counterpart in at least one embodiment is the fact that it replicates the identical motions that a user who is making a copy using a physical photocopier goes through. When a user photocopies a document, he/she selects where they want to copy from (i.e. the sheet feeder), where the user wants to copy to (i.e. 6 copies collated and stapled) and then presses a GO button to actually carry out the photocopy process. With Virtual Copier the process feels familiar because the sequence is the same with just the Power VC portion of the main Virtual Copier window.

The power of Virtual Copier is the fact that the From can be a physical device (e.g. digital copier, fax or scanner) or an application (e.g. Lotus Notes, Microsoft Exchange, the Internet, or an electronic filing system). The To can also be a physical device (e.g. a fax, digital copier, or printer) or an application (e.g. Lotus Notes, Microsoft Exchange, the Internet, or an electronic filing system). Even though paper is being copied electronically from devices to applications, from applications to devices, from devices to devices, or from applications to applications, the user simply has one sequence to execute: select From, select To, and then press GO. Virtual Copier will accomplish all translations between device and applications automatically and seamlessly.

Another reason that paper is still a major corporate issue is that traditional document management systems require that a company invest in a whole new system just to store electronic images. Although this is the only way that document management systems have been designed and delivered, it is in fact highly inefficient. Most companies already manage information about physical documents in some form of software applications.

For example, accounting systems have long been used to maintain information about invoices and bills that arrive into a company from outside sources as physical pieces of paper. When an invoice arrives, its information is keyed into the accounting software, where balances are maintained and accounts payable information is coordinated. Yet the original invoice is stored manually, and every time that a request is made for a copy of the signed invoice, someone manually retrieves the invoice from a physical filing cabinet. Accounting systems, like most business applications, typically have no way of maintaining an electronic copy of the physical invoice, and adding a document management system to an accounting system is cumbersome, costly, and difficult to maintain, and even more difficult to coordinate.

Virtual Copier solves this problem in at least one embodiment by copying paper directly into the existing accounting system. Simply adding a To item in the Virtual Copier window enables a user to copy paper directly into the appropriate accounting record of the existing accounting system. This requires no retraining (users who are trained on the accounting system will still use the accounting system in the same way), requires no document management system (the electronic copy of the document is actually being maintained by the accounting system itself), there is no coordination between two systems (Virtual Copier embeds the invoice with the appropriate accounting record), and it is simple (one Go button).

What is true with regard to the example above of an accounting system is true of most other business applications. The power of Virtual Copier is that it can turn an information system into a document management system by adding support for electronic paper directly into the existing business application, whether it is a client, server-based, or web-based system.

Virtual Copier enables corporations to perform sophisticated document imaging with their existing Web-based and client-server applications through a user interface that is as familiar as the office copier. Virtual Copier can be used out-of-the-box as a standalone application to copy, scan, fax, or print images using existing digital devices within corporate environments or across the web. With the extensions, as described below, Virtual Copier can be integrated into Web-based and client server applications, such as ERP or accounting systems, to eliminate paper from existing business processes and legacy applications. Virtual Copier can also be used to support seamless access to document image processing and archival over the web since, in at least one embodiment, the VC interface is implemented as a software application.

VC is architected as an application that delivers end-user functionality while remaining open to third-parties extensions. For example, VC can be viewed as a copier. Like a copier, VC takes paper in, and produces paper going out. The only difference is that VC does not distinguish between electronic and physical paper.

To accommodate third-party extensions, VC is divided into five essential modules. Each module is a counterpart to an aspect that is found on a conventional copier. Based on the modular design of VC, each aspect of VC can be independently extended, offering much greater flexibility than conventional copiers.

The five core modules of VC illustrated in are:

Input Module—The Input Module manages paper or electronic paper entering VC. This module manages imaging devices to input paper through scanners, MFPs, or the new breed of digital copiers. The Input Module also manages reading electronic paper from third-party or proprietary applications. The counterpart to VC's Input Module on a conventional copier is the scanner subsystem.

Output Module—The Output Module manages paper or electronic paper exiting VC. Like the Input Module, this module manages imaging devices to output paper to standard Windows printers, specialty image printers, MFPs, or the new breed of digital copiers. The Output Module also manages writing electronic paper to third-party or proprietary applications. The counterpart to VC's Output Module on a conventional copier is the printer or fax subsystem.

Process Module—The Process Module applies processing to the electronic paper as it is being copied. Examples of a process are OCR and ICR. The Process Module can also apply non-imaging functionality as well, such as workflow or other relevant tie-ins to the electronic paper as it is being copied. One of the advantages of VC over conventional copiers is that multiple processes can be applied to a single virtual copy. The counterpart to VC's Process Module on a conventional copier is the controller.

Client Module—The Client Module presents the electronic paper as it is being copied, and any relevant information related to the input or output functions. For example, if the Output Module is directed to a printer, then the Client Module might present the finishing capabilities; if the Output Module is directed to Goldmine, then the Client Module might present the target contact record to which the document is being copied. The counterpart to VC's Client Module on a conventional copier is the panel.

Server Module—Unlike conventional copiers, VC's Server Module is a unique subsystem that can communicate with the other modules as well as third-party applications. The Server Module is what makes VC a far more powerful concept than simply an application that can control a scanner and a printer to mimic a copier. The Server Module can be used to combine third-party applications with the new breed of digital imaging devices to create unique and custom virtual copier solutions. A virtual copier can be created with VC by combining a scanner with a printer; or by combining a scanner with an application; or by combing an application with an image printer. In each case VC is dynamically creating a custom virtual copier, with a complete understanding of how paper flows from the source to its destination. There is no counterpart to VC's Server Module on a conventional copier.

One of the primary design goals of VC is to make it simple to integrate VC with third-party applications. There are two options to integrating VC into a third-party application: running VC as an external service, or embedding VC as an underlying service.

VC is in one embodiment and optionally a standalone application that enables a user to scan (copy) paper from a device to a third-party application, and to print (copy) the reference of an image document from a third-party application to a printing device. VC does not require the third-party application to be aware that VC is operating. Rather, VC recognizes that the third-party application is running, and it intelligently copies paper to and from that application.

In this scenario the user is interacting with VC's Client Module in order to execute a copy operation to and from the third-party application. There does not have to be any changes made to the third-party application, not even to its interface, in order for VC to operate. The user of VC only knows that to copy to and from the third-party application, a custom Input and Output Module must be selected, and the Go button is pressed.

In order to support copying to and from a third-party application, VC must be able to support extensions that understand each third-party application. This is accomplished through the Input and Output Modules. The Client, Server, and even Process Modules remain independent across third-party applications. However, in order to support outputting to a third-party application, an Output Module is developed that is unique to that third-party application. Likewise, an Input Module is developed that is unique to a third-party application in order to support reading images from that application.

It is the optional Input and Output Modules that render VC extendable. For each third-party application there is a unique pair of Input and Output Modules that understand the third-party application, and how to copy images to and from that application. Each Input and Output Module registers itself to the Windows registry so that the Server Module knows how to find them. In this way Virtual Copier can grow indefinitely, to support any number of third-party applications.

The significant point is that the Input and Output Modules have their own interface, and can be developed independently from any other module. As long as the Input and Output Module conform to the API specified in this document it will plug-and-play with VC. VC will be able to mix and match the custom Input and Output Module with its standard and other custom Input and Output Modules.

A third-party application can also use the services of VC without its user interface. That is, a third-party application can embed VC's functionality and provide its own interface to its functionality. For example, rather than have VC as a separate application, a special button can be placed on a third-party application that launches VC in the background.

VC is designed so that the Server Module can run independently from the Client Module. All the core functionality, including communicating with the Input, Output, and Process Modules, are performed directly by the Server Module. The Client Module is generally simply an interface to the Server Module. Therefore, all the services of the Server Module can be made available in the background to a third-party application without the need for an interface. The third-party application can in fact become the user's interface to VC.

In order to support VC operating in the background a third-party application merely has to communicate with the Server Module directly, as described later in this document. The Server Module, as all modules in VC, support COM-based interfaces for simple and direct support from all major Windows development environments.

The purpose of the computer architecture and process described herein is to create a component factory that can automatically generate reusable software components from sophisticated core software technologies. Many, if not most, core software technologies, such as OCR (Optical Character Recognition) or barcode recognition, are designed and implemented using a "C"-language API (Application Program Interface). The technology is often complex, requiring months of trial-and-error to correctly develop application systems using the technology. While there are millions of Intranet developers and power-PC users who are capable of assembling component-based systems, I have determined that there are relatively few "C" programmers (estimated at less than 100,000) who can learn and implement application software with these complex __C"-level API's. It is therefore desirable to develop software tools for automatically generating reusable software components from core software technologies thus making these software technologies available to a much larger user base.

Since I have determined that there is no structure or format for implementing "C"-level API's, the ability to automatically transform a unique API into a standard component would seem impossible since that would take a nearly-human level of intelligence. To date, the only way, I am aware, to create a component out of an existing API is to have an existing programmer in the field do the work for each API. Humans can intelligently analyze an API and create a component based on intelligent decisions tempered by experience. The challenge of creating a component factory is the challenge of partially or substantially recreating the component design and formulating effective implementation decisions.

One would expect the translating a "C"-level API from its native state into a component would require human-level intelligence. This is mainly because "C"-level APIs have virtually no constraints as to how they can be implemented. This means that there are an infinity variations of APIS, which can only be managed by human-level intelligence. While this point is true, I have determined that the appropriate solution starts at the other side of the equation, which is the component itself.

My solution starts out with a definition of a component that can sustain the feature/function requirements of any API. In other words, the interface of a generic component can be defined such that the features and functions of virtually any API can be re-implemented within its bounds. The two known end-points are the "C"-level API that started with, and the component interface that represents any set of features/functions on the other side.

I have also determined that one solution for creating a computer architecture and process for implementing a component factory is to create a well-defined multi-tiered systems architecture for a component and to automate, substantially automate, or manually expedite from its native state through the various tiers of the systems architecture resulting in a standardized or substantially standardized component. Advantageously, this solution is not based on making human-level intelligent decisions on how to translate a _C"-level API into a component. Rather, by starting with a well-defined systems architecture that is multi-tiered, a series of incremental steps that migrates a C-level API from one tier within the systems architecture to the next may be performed, and which are facilitated using the architecture and/or process described herein.

Advantageously, each incremental step is not a major one, but in sequence the entire series of steps will result in a usable component. Since each step of migration is not a major one, the chances of automating these steps is significantly higher and the likelihood of being able to create the component factory becomes more feasible.

The fundamental building blocks of the computer architecture and process are twofold:

1) To define a systems architecture that describes in detail how to implement a component from a C-level API
2) To create a component factory by automating, substantially automating, or manually expediting the migration of a C-level API from one tier within the architecture to the next.

The building blocks are the keys or important to actually making the component factory feasible.

Significantly, the computer architecture and processes described herein have application to the Intranet and document market marketplace. Corporations are embracing internet computing technologies to create enterprise-level Intranets and Extranets. Using standard browser technologies, corporations and government entities are rapidly adopting the internet computing model and are developing enterprise applications by assembling standard Microsoft specified Active X components. These are not "C" programmers; rather they are typical power PC users. Further availability of reusable components would only fuel this development.

The general outline for creating a component factory is described below in detail. It is important to note that automatically, substantially automatically, or manually building a component is neither obvious nor guaranteed. As will be described below in detail, automating or substantially automating the building of a component consists of automating individual steps that comprise the component architecture. However, in today's application environment, any amount of automation will dramatically increase the efficiencies of building a component The computer architecture is designed for managing a diverse set of independent core technologies ("engines") using a single consistent framework. The architecture balances two seemingly opposing requirements: the need to provide a single consistent interface to many different engines with the ability to access the unique features of each engine.

The benefit of the architecture is that it enables a company to rapidly "wrap" a sophisticated technology so that other high-level developers can easily learn and implement the core technology. The computer architecture is therefore a middleware or enabling technology, as illustrated in FIG. 1.

As illustrated in FIG. 1, computer architecture 2, described below in detail, is a middle layer between high level developer programs 4 (such as C-level APIs, or other programs having similar characteristics) and are technology/component engines 6 (such as OCR, bar code recognition, and other components having similar characteristics).

Another benefit of the architecture is that it provides a high-level specification for a consistent interface to any core technology. Once a high-level developer learns the interface described herein for one engine, that knowledge is easily transferable to other engines that are implemented using the architecture. For example, once a high-level developer learns to use the computer architecture for OCR (Optical Character Recognition), using the computer architecture for other engines, such as barcode recognition or forms processing, is trivial.

In summary, the architecture and process described herein is at once a framework for rapidly wrapping sophisticated technologies into high-level components, as well as a framework for high-level developers to communicate with a diverse set of engines. The creating of a component factory is based on the fact that the architecture defines a clear path for "wrapping" any C-level API into a component using simple structures and many rote steps. This process is currently being done in an inefficient manner by a programmer in the field.

The method described herein for creating a component factory creates a well-defined multi-tiered architecture for a component and automates, substantially automates, or manually expedites (hereinafter "automates") the process of migrating a "C"-API from its native state through the various tiers of the architecture resulting in a standardized component.

Advantageously, the method described herein does not base the component factory on making human-level intelligent decisions on how to translate a "C"-level API into a component. Rather, by creating a well-defined architecture described below that is multi-tiered, the method is a series of incremental steps that need to be taken to migrate the "C"-level API from one tier within the architecture to the next. In this way each incremental step is not a major one, but in sequence the entire series of steps will result in a component.

Since each step of migration is not a major one, the chances for automating these steps is significantly higher and the likelihood of being able to create the component factory becomes feasible. This approach is in fact what makes the method cost-effective, since the alternative approach, i.e., computer-generated human-level decision making, is currently unavailable and would require much effort, if at all possible, to replace humans in any realistic decision-making process.

With a fixed architecture that can be used to implement a "C"-level API as a component (using a programmer), that same architecture can be used as the basis for the component factory model. In order to make the component factory, each step of the architecture needs to be designed to facilitate automation or manually expedited. In other words, I have determined that automating/expediting the process of taking the original "IC"-level API and migrating it to a Level 1 layer, and then a Level 1 to a Level 2, and then a Level 2 to a Level 3 layer, and so on, the component has been implemented automatically, or more efficiently manually. The component factory is therefore a sum of the ability to automate migrating the "C"-level API from one layer to the next within a well-defined architecture for implementing components.

As illustrated in FIG. **2**, the component factory **10** migrates the original "C"-level API **12** from its original state into the generic interface **8** defined by the topmost layer. The first feature that can be demonstrated is that there is a topmost layer **8** that can define a component interface that can represent the features/functions of most core technologies. The component factory **10** migrates the "C"-level API **12** to the topmost level 8. Doing this in one large step would be impossible since the "C"-level API has a near-infinite variety of styles. However, the architecture advantageously has enough well-defined and well-structured layers for implementing the topmost component interface, for creating the component factory.

A simplified overview of the architecture is illustrated in FIG. **3**. In FIG. **3**, the component interface **8** sits on top of an Object Manager **14** that communicates with individual objects e.g., **16**, **18**, **20**. These objects **16**, **18**, **20** represent specific core technologies that are represented as "C"-level APIs. The design of Object**1**, Object**2**, . . . ObjectN is illustrated in FIG. **4**.

A component factory can be created by automating the process of migrating the original "C"-level API **12** from its original state to the Layer **1**—Engine Management tier **26**, and then from the state to Layer **2** Engine configuration tier **24**, and so on up the Engine Functions layer **22**. These layers will be further described below.

The computer architecture is implemented, for example, as a standard COM component, as an ActiveX control; the specifications designed by Microsoft, published in the technical literature, and incorporated herein by reference. ActiveX control (COM) support is currently available within any Microsoft 32-bit Windows operating environment. ActiveX controls are supported by all OLE-based applications, including all of Microsoft's end-user products (e.g., Microsoft Office, Word, Access, Powerpoint, Access), the main Internet Browsers (Microsoft's Internet Explorer and Netscape's Navigator—the latter with an add-in product and by 4Q97 directly), most other name-brand end-user Windows products (e.g., Lotus Notes), and all major development environments (e.g., Microsoft Visual Basic and Visual C++, Delphi, Borland C++, Power Builder). By implementing the architecture as, for example, an ActiveX control, complex technologies can be programmed by virtually any Windows or Intranet user or developer. Of course, other component specifications may also be used.

Although the architecture has been implemented as a COM-based technology with C++ as the language of choice, the architecture can be implemented in many other languages (e.g. Java) and distributed architectures (e.g. CORBA).

Every engine, such as a text retrieval or an OCR (Optical Character Recognition) engine, has a unique interface. This interface is generally a "C"-level API (Application Program Interface). In most cases, the learning curve for understanding and integrating a new engine can be a one man-month to several man-years and generally requires highly experienced "C" programmers. The purpose of the architecture is to define a clear infrastructure within which any core can be rapidly "wrapped" so that users and developers can have easy access to these core technologies.

In addition to defining the infrastructure for engines to be accessible to typical users, the architecture also defines how to migrate an engine from its native state to the prescribed interface. In other words, the architecture goes beyond simply defining the framework for wrapping engines, it also defines the specific steps for wrapping these engines.

The architecture consists of a hierarchical series of layers that take any "C"-level API from its unique state to one that is standard and consistent. The result is a single, highly-integrated object component that contains and manages any type of engine that can be programmed regardless of the nature and subject of the core technology. The architecture therefore not only defines the goal (e.g., the object component interface) but also the means of implementing that goal for any type of engine.

The architecture is comprised of two major parts as illustrated in FIG. **5**: the Object Manager **14**, and the individual object components **16**, **18**, **20**. The Object Manager **14** in FIG. **5** manages individual object components **16**, **18**, **20** illustrated as Object **1**, Object **2**, etc. The Object Manager **14** communicates with the individual object components **16**, **18**, **20** using a consistent COM interface.

Each object component implements the feature set of an individual engine by mapping a consistent COM interface to the "C"-Level API interface of the individual engine that it supports. In this way the Object Manager can consistently communicate with each engine, using the engine's object component. Because the COM interface of each object component is consistent, the Object Manager can interface with every underlying engine the same way.

The features of the architecture include:

1) definition of consistent COM interfaces for individual object components that represent diverse technologies;

2) a prescribed process for migrating any engine to the defined consistent COM interface; and/or

3) a predefined Object Manager that automatically manages the individual object components.

When implemented, for example, as an ActiveX control, the architecture also yields an umbrella control that can be used by a high-level programmer to program and manage numerous sophisticated technologies in a plug-and-play environment. In order to facilitate the discussion of the architecture itself it is best to start with the architecture of the engine object component and then describe the Object Manager. Since the Object Manager is directly dependent on the engine object components, an understanding of the latter will assist in the description of the former.

Engine Object Component—**16**, **18**, **20**

The purpose of the engine object is to wrap a specific engine using a series of layers that convert the engine's unique interface into a COM interface that is, for example, specified by the architecture. The architecture not only defines the consistent COM interface for implementing an engine, it also describes how to implement the interface from the original "C"-Level API. Once the COM interface of the engine object component is implemented, the Object Manager understands and can therefore communicate with it.

Each engine component consists of, for example, three layers that are designed to migrate the original API of the engine to a consistent COM interface. As illustrated in FIG. **6**, the Object Manager **14** communicates with the topmost layer **22** of the object component **16**, **18**, **20** which is the defined interface of object component.

Each layer is described below in two parts. The first part is the prescribed COM interface for communicating with the engine object component. The second part describes a specific path for automating building the layer. By providing an outline for automating building each layer, the overall engine object component can be automatically, substantially automatically or manually expedited and generated.

Layer 1—Engine Management 26

The first layer in the object component architecture is designed to deal with the fundamental features of an engine. This includes the ability to load and unload the standard or commercially available via, for example, MicroSoft Corporation, engine Dynamic Link Libraries (DLLs) into memory, as well as the ability to consistently deal with errors. This is the most fundamental layer because it is the essential "wrapper" layer of an engine. Once this layer is complete all interaction with the underlying engine is filtered through this layer. Additional important engine management functions include dynamically accessing a function call of an engine, and initializing engine settings. All of these engine management functions are optionally and beneficially table driven to promote or facilitate access to, and implementation of, engine management functions.

The Layer 1 specification is summarized in FIG. 7 that describes the IEngineManagement COM interface. The purpose of the IEngineManagement interface is to transparently load and unload an engine to and from memory. I have determined that this is often the core feature that is incorrectly implemented and a cause for hard-to-find bugs. This layer may be generated manually by a developer who is familiar with the architecture as outlined herein in an expedited manner or automatically as described below in detail.

Layer 1 can be precisely defined in generic terms, and is therefore the simplest layer to likely be automatically, substantially automatically, or easily manually generated. A sample or example of actual code that can be used to implement this layer is described below. As long the process and/or code for implementing Layer 1 can be generically defined, that is engine and technology independent, then the process of generating the generic code for each new engine is expedited either manually or automatically.

The premise for automating any level is to start with as few pieces of information as possible. For the Engine Management layer 1 have assumed that nothing more than the set of DLLs that implement the engine functionality are known. Given this information, I have determined that I will need to implement:

Loading and unloading the engine from memory

Adding error management We can start, in this example, with a model C++ header file that defines the Engine Management layer and investigate how this code can be implemented generically.

As mentioned earlier, if the code to implement this layer can be defined generically then it can be easily generated, for example, manually, and/or automatically for any engine.

```
class SomeEngineObject
{
      //Wrapper Functions
private:
      FARPROC_SomeFunction;
      BOOL SomeFunction○;
      //EngineManagement
protected:
      BOOL GetProcAddress (HINSTANCE, FARPROC&, LPCTSTR);
      BOOL GetProcAddresses○;
      BOOL ProcessError○;
```

-continued

```
public:
      BOOL ActivateEngine (BOOL Activate);
      BOOL IsEngineActivated○;
};
```

The IEngineManagement interface is implemented in the C++ class as the public methods: ActivateEngine○ and IsEngineActivated○.

The first step of implementing the Engine Management layer **20** is to wrap each original engine function within a class-defined function that represents the original. For example, if there is an original function called SomeFunction○, then the engine object should have a corresponding SomeFunction○ method. The engine object version can then add standard engine and error management code so that any layers above have automatic error detection, correction, and reporting.

An example of generic code that maps an original function call to the original function is as follows:

```
BOOL GetProcAddress (HINSTANCE hLib, FARPROC&Proc,
LPCTSTR ProcName)
{
      Proc= :: GetProcAddress (hLib, ProcName)
      if (!Proc)
      {
            SetIMAGmEError (LOADENGINEFUNCTIONSERROR,
ProcName);
            return FALSE;
      }
      return TRUE;
```

Given the original function name, the GetProcAddress can map the original function to one that is defined by the engine object. Using the engine object C++ header file described above, the SomeFunction○ method is mapped to the original engine function using the following line of code: (GetProcAddress (hLib, SomeFunction, "SomeFunction");

To map all the function calls within the original engine DLLs just requires cycling through each function call and mapping it to the engine object counterpart. Since Windows contains facilities that enables access to all the functions within a DLL, a simple loop may be used. The hLib module is derived from the DLL name, which, as mentioned at the start, is the one piece of information we are given.

What is more complex is to define a generic implementation of the engine object version of the original function. This may be described in code as follows:

```
                  BOOL SomeFunction (arguments)
                  ASSERT (arguments)
                  ErrorVariable=_SomeFunction (arguments);
                  returnProcessError○;
            }
```

The engine object version of the original function passes the function call to the original one after completing a series of assertion tests, and is followed by a series of error detection tests. In this way the original engine function is "wrapped" by the engine object to manage error detection and correction.

The process of loading an engine can likewise be implemented generically.

```
BOOLLoadDLLs○
{
    BOOLbReturn=TRUE;
    HINSTANCEt_hLib;
    CStringt_ModuleName;
    POSITIONpos;
    pos=m_Modules.GetStartPosition○;
    if (pos=NULL)
    {
        SetIMAGinEError (NOMODULESDEFINED);
        return FALSE;
]
    while (pos&&bReturn)
    {
        m_Modules.GetNextAssoc (pos, t ModuleName, t hLib);
        if (t_hLib!=NULL)
            continue;
        t_hLib=::LoadLibrary(t_ModuleName);
        if (t_hLib=NULL)
        {
            SetIMGAinEError (CANTLOADMODULE,
            t_ModuleName);
            FreeDLLs○;
            bReturn=FALSE;
            break;
        }
        m_Modules.SetAt (t_ModuleName, t_hLib);
    }
returnbReturn;
}
```

The LoadDLLs function is a generic implementation of a function that loops through the names of DLLs that are provided (in the form of the mModules variable), and cycles through each one loading it into memory using the Windows LoadLibrary○ function. A similar engine object function can be implemented to remove these DLLs from memory.

The present invention further divides the engine management layer into three functions, as illustrated in FIG. **8**. The first function is loading and unloading **124** of the core or engine technology. The second function for the engine management layer **26** is dynamically linking procedures or function calls, or hooking the desired engine functionality into the procedures of the core technology **126**, including, for example, initializing and setting up engine settings. The third function is initializing the engine itself **128**, which is essentially engine management. Once these three functions are performed in level **1**, anything in the core technology is accessible.

Advantageously, the present invention utilizes tables to drive each of these three functions described above, and as illustrated in FIG. **9**. Each of the tables of files, for example tables **130**, **136**, **140**, are filled in with the appropriate data or information. I have discovered that if the above three functions are set up or implemented using tables, that the core technology may be effectively and efficiently described. That is, the use of tables is a very effective and simple method of describing an engine for use in engine management, engine loading/unloading and engine procedure linking. For example, it is similar to indicating or providing the raw data of that engine, the list of the engine functions, and the list of the engine dynamic link libraries (DLLs) for engine management.

The files or tables contain the logic or executable of the engine. Accordingly, all that is needed is a list of the engine functions **132**, a list of the file of the engine executable code or DLLs **138**, and a list of the engine settings **142**. Using the tables with the above information, the engine may be automatically loaded and unloaded, initialized, and/or dynamically hooked into the necessary functions. Accordingly, the process of generating level 1 for engine management may advantageously be automated. The specific algorithms used for the engine management layer are described in the Appendix.

In summary, for the Engine Management layer the following pieces may be automated, substantially automated, and/or manually expedited.

Loading and unloading the engine DLLs (provided into and out of memory

Mapping original functions to engine object counterparts

Adding general error detection and correction

Determining and matching arguments and return values for mapping the original functions to their engine object counterparts. In order to add assertion and error detection and correction, the original function must be wrapped and called from within the engine object version of the original function.

Managing error feedback. All APIs have their own way of providing error feedback. Since one of the goals of the Engine Management layer is to generically manage error detection, correction, and feedback, it must handle all errors identically. However, APIs have numerous and incompatible methods in this case. I have determined that most APIs follow one of several distinct mechanisms for providing error feedback.

By creating specific classes of APIs, the process of generating Layer **1** engine management may be expedited, manually and/or automatically.

Layer **2**—Engine Configuration **24**

The second layer **24** in the object component architecture is designed to deal with configuring an engine. This includes the ability to set any variety of features that are generally associated with the functioning of an engine. The architecture is designed to meet the challenge of providing a uniform interface for dealing with generally any or most engine settings.

The engine configuration layer **24** includes a series of prefabricated functions that map out the settings stored in the table to the appropriate engine configuration parameters. Accordingly, all that is needed is to fill in the values for the table associated with engine configuration. Thus, the engine object may advantageously come pre-packaged with predetermined tables populated with predetermined values.

The Layer **2** specification can be summarized in FIG. **10** that describes an exemplary IEngineConfiguration COM interface. The purpose of the IEngineConfiguration interface is to provide the ability to set and get the settings of any engine uniformly. While the Engine Management layer can load and unload engines transparently, this layer configures engines to operate as required by the user or developer.

FIG. **11** is another exemplary table illustrating the engine configuration specification. Examples include a set setting function **144**, a get setting function **146**, a load setting **148**, a save setting **150**, an is setting valid function **152**, a default setting **154**, and a prompt setting **156**.

The get setting **146** and set setting **144** functions retrieve the value of a particular engine setting, or assign a value to a particular engine setting, respectively. Each one of the get setting and set setting functions includes or comprises a table of the settings. The load setting **148** and save setting **150** functions do the similar function as the get setting and set setting functions, but in persistence. Persistence is defined as writing values to the disk, for example hard disk, compact disk, and the like, and retrieves the values from the disk. So as where the get setting and set setting functions assign a value and/or retrieves the value from local memory, the load and set setting functions assign the value and retrieve the value of the setting from disk.

The load and set setting functions provide persistence when the computer system is close down, such that when the computer system will return to the last setting when it is subsequently reopened.

The default setting function **154** provides the most favorable value for a given setting. Thus, if no setting is selected, the system will automatically select default settings. The prompt setting function **156** is what displays to the user all the various settings.

Advantageously, the present invention generates the skeletal structure of each table automatically. In addition, since there is a table of settings, the skeletal structure not only generates these functions, but also fills in the settings that need to be assigned. Thus, the engine configuration function provides the feature of having a pre-populated set of options which require particular values to be assigned to table entries.

Although this architecture advantageously makes it simple for a human to migrate the configuration of an engine appear into two simple and universally applicable interface points, doing so automatically requires additional steps. The two steps to automating this approach are, for example, as follows:

Determine the configuration methods used by various APIs for configuring the core technology;

Detect the variations for configuring an engine and automating each one separately.

As with Layer **1**—Engine Management, there exists a finite set of general variations used by developers of core technologies to configure an engine. Although Layer **1** is clearly more generic in nature, advantageously, Layer **2** also has considerable consistency.

Layer 3—Engine Functionality 22

The third layer **22** in the object component architecture is designed to deal with accessing the actual functionality of the core engine. For example, for an OCR engine this would be to OCR an image or a document. For a text retrieval engine this would be to initiate and retrieve results of a text search.

An exemplary Layer **3** specification can be summarized in FIG. **12** that describes the IEngineFunction COM interface. The purpose of the IEngineFunction interface is to provide the ability to initiate any function supported by an engine. The simple IEngineFunction interface is capable of managing an infinite variation of functions.

The third layer may advantageously be further divided into many sub-layers that more discretely define the steps necessary to execute a function within an API. Since the designer of an API has infinite variety of possible ways of implementing a function, creating a tiered architecture to manage this layer is useful.

An exemplary tiered architecture for the engine function is illustrated in FIG. **13**. As illustrated in FIG. **13**, the engine function or engine processing layer includes four elements. The engine function layer **22** includes a series of predefined functions to perform in the perform element **158**. For example, for optical character recognition (OCR), the present invention uses a set of predefined functions. Alternatively, for scanning, the present invention includes a separate set of predefined functions.

Accordingly, there are a series of actions that are performed by the engine function layer on a given engine, such as an OCR engine, a scanning engine, a printing engine and the like. The engine function layer is designed not to generally go directly to a specific engine. Rather, the engine function layer **22** will generally interface with the engine management layer **26** and/or the engine configuration layer **24** as needed.

For example, in the course of performing an action and/or function, the engine function layer interfaces with the engine configuration layer to possibly modify settings. For an OCR engine, the engine function layer fills out a table of OCR documents as one action that could take place. OCR image is another action.

The get function results **160** gets the results of the function stored in a register. The clear function **162** clears all the registers that contain all the results, in this case its memory. The feedback event or function **164** provides continuous feedback, depending on what action takes place. For example, if an OCR action is being performed, the feedback function provides the percentage of completion of the OCR process.

The automation of this layer is accomplished by the following functions:

Determine the execution of methods used by various APIs for executing a given function;

Divide this layer into a multi-tiered layer that further facilitates automation;

Detect the variations of the sub-layers and automate each one separately.

Although this layer has many more variations than Layer **2**, I have determined that there is a general set of variations used by developers of APIs to implement core functionality.

Thus, the benefit of the component factory is that it can transform core software technologies that are currently available in "C"-level APIs to a limited audience into components that have a much greater audience.

There are a variety of "C"-level APIs that cover the following categories of functionality that can be better served in the market as ActiveX controls or other component and used in conjunction with the architecture and methods described herein.

Text Retrieval

Data Extraction

Workflow

Storage Management

Each of these categories has several vendors with products that currently service the market in a limited way because the technologies are only available as "C"-level APIs. Without the core competency of creating components out of these core technologies they are limiting their marketability and opportunity for international distribution.

With the proposed component factory users and vendors can rapidly create components from their original core technology and increase their marketability, competitiveness, and ultimately their sales.

Further, there are numerous core technologies, such as text-retrieval and ICR (Intelligent Character Recognition), that have already been implemented, and are only available as "C"-level APIs. Many, if not most, core technologies are first released exclusively as "C"-level APIs. While there are integrators and corporations who have the team of technologists who can integrate these "C"-level APIs in-house, most companies are looking for component versions that can be implemented at a much higher level. Therefore, many of the core technologies that are only available in a "C"-level API are not being used due to their inaccessible interface. The benefit of the component factory is that it can rapidly make available core technologies implemented as "IC" APIs that would otherwise be underutilized or dormant in research labs by converting them to high-level components that can be used by millions of power-PC users.

With the advent of the World Wide Web (WEB) this opportunity has increased exponentially. The WEB is now home to a vast number of WEB authors with minimal formal training who can implement HTML pages and build web sites. One of the fundamental technologies for extending the capability of the WEB from simple page viewing to interactive and sophisticated applications is components. A component extends the capability of HTML by enabling a WEB author to add core technology as a pre-packaged technology. Since components are fundamental to the growth and usability of the WEB, having a component factor that can translate "C"-level toolkits into components that are then usable within WEB sites opens a vast and new worldwide market to these technologies.

FIG. **14** is an illustration of a main central processing unit for implementing the computer processing in accordance with a computer implemented embodiment of the present invention. The procedures described above may be presented in terms of program procedures executed on, for example, a computer or network of computers.

Viewed externally in FIG. **14**, a computer system designated by reference numeral **40** has a central processing unit **42** having disk drives **44** and **46**. Disk drive indications **44** and **46** are merely symbolic of a number of disk drives which might be accommodated by the computer system. Typically these would include a floppy disk drive such as **44**, a hard disk drive (not shown externally) and a CD ROM indicated by slot **46**. The number and type of drives varies, typically with different computer configurations. Disk drives **44** and **46** are in fact optional, and for space considerations, may easily be omitted from the computer system used in conjunction with the production process/apparatus described herein.

The computer also has an optional display **48** upon which information is displayed. In some situations, a keyboard **50** and a mouse **52** may be provided as input devices to interface with the central processing unit **42**. Then again, for enhanced portability, the keyboard **50** may be either a limited function keyboard or omitted in its entirety. In addition, mouse **52** may be a touch pad control device, or a track ball device, or even omitted in its entirety as well. In addition, the computer system also optionally includes at least one infrared transmitter **76** and/or infrared receiver **78** for either transmitting and/or receiving infrared signals, as described below.

FIG. **15** illustrates a block diagram of the internal hardware of the computer of FIG. **14**. A bus **56** serves as the main information highway interconnecting the other components of the computer. CPU **58** is the central processing unit of the system, performing calculations and logic operations required to execute a program. Read only memory (ROM) **60** and random access memory (RAM) **62** constitute the main memory of the computer. Disk controller **64** interfaces one or more disk drives to the system bus **56**. These disk drives may be floppy disk drives such as **70**, or CD ROM or DVD (digital video disks) drive such as **66**, or internal or external hard drives **68**. As indicated previously, these various disk drives and disk controllers are optional devices.

A display interface **72** interfaces display **48** and permits information from the bus **56** to be displayed on the display **48**. Again as indicated, display **48** is also an optional accessory. For example, display **48** could be substituted or omitted. Communication with external devices, for example, the components of the apparatus described herein, occurs utilizing communication port **74**. For example, optical fibers and/or electrical cables and/or conductors and/or optical communication (e.g., infrared, and the like) and/or wireless communication (e.g., radio frequency (RF), and the like) can be used as the transport medium between the external devices and communication port **74**.

In addition to the standard components of the computer, the computer also optionally includes at least one of infrared transmitter **76** or infrared receiver **78**. Infrared transmitter **76** is utilized when the computer system is used in conjunction with one or more of the processing components/stations that transmits/receives data via infrared signal transmission.

FIG. **16** is a block diagram of the internal hardware of the computer of FIG. **14** in accordance with a second embodiment. In FIG. **16**, instead of utilizing an infrared transmitter or infrared receiver, the computer system uses at least one of a low power radio transmitter **80** and/or a low power radio receiver **82**. The low power radio transmitter **80** transmits the signal for reception by components of the production process, and receives signals from the components via the low power radio receiver **82**. The low power radio transmitter and/or receiver **80**, **82** are standard devices in industry.

FIG. **17** is an illustration of an exemplary memory medium which can be used with disk drives illustrated in FIGS. **14**–**16**. Typically, memory media such as floppy disks, or a CD ROM, or a digital video disk will contain, for example, a multi-byte locale for a single byte language and the program information for controlling the computer to enable the computer to perform the functions described herein. Alternatively, ROM **60** and/or RAM **62** illustrated in FIGS. **15**–**16** can also be used to store the program information that is used to instruct the central processing unit **58** to perform the operations associated with the production process.

Although processing system **40** is illustrated having a single processor, a single hard disk drive and a single local memory, processing system **40** may suitably be equipped with any multitude or combination of processors or storage devices. Processing system **40** may, in point of fact, be replaced by, or combined with, any suitable processing system operative in accordance with the principles of the present invention, including sophisticated calculators,and hand-held, laptop/notebook, mini, mainframe and super computers, as well as processing system network combinations of the same.

Conventional processing system architecture is more fully discussed in *Computer Organization and Architecture*, by William Stallings, MacMillam Publishing Co. (3rd ed. 1993); conventional processing system network design is more fully discussed in *Data Network Design*, by Darren L. Spohn, McGraw-Hill, Inc. (1993), and conventional data communications is more fully discussed in *Data Communications Principles*, by R. D. Gitlin, J. F. Hayes and S. B. Weinstain, Plenum Press (1992) and in *The Irwin Handbook of Telecommunications*, by James Harry Green, Irwin Professional Publishing (2nd ed. 1992). Each of the foregoing publications is incorporated herein by reference. Alternatively, the hardware configuration may be arranged according to the multiple instruction multiple data (MIMD) multiprocessor format for additional computing efficiency.

The details of this form of computer architecture are disclosed in greater detail in, for example, U.S. Pat. No. 5,163,131; Boxer, A., Where Buses Cannot Go, IEEE Spectrum, February 1995, pp. 41–45; and Barroso, L. A. et al., RPM: A Rapid Prototyping Engine for Multiprocessor Systems, IEEE Computer February 1995, pp. **26–34**, all of which are incorporated herein by reference.

In alternate preferred embodiments, the above-identified processor, and in particular microprocessing circuit **58**, may be replaced by or combined with any other suitable processing circuits, including programmable logic devices, such as PALs (programmable array logic) and PLAs (programmable logic arrays). DSPs (digital signal processors), FPGAs (field programmable gate arrays), ASICs (application specific integrated circuits), VLSIs (very large scale integrated circuits) or the like.

FIG. **18** is an illustration of another embodiment of the component factory migrating the original "C"-level API from its original state into the generic interface defined by the topmost layer. This powerful architecture goal is to supply easy access to all imaging functions that can be performed by any engine.

The architecture according to this second embodiment, groups C-level toolkits **100** into logical categories, such as scan, print, display, OCR, cleanup and so on. A single engine can span multiple categories (e.g., Kofax engine does view/print/scan). This enables the architecture to deal with the multitude of engines available in a logical fashion.

On top of these, a three-level C++ class (or object) **102** is built for each engine. This object gives uniform access to the engine and to all its unique settings. The three levels do the following:

Level 1 of the C++ classes **112** is a protective wrapper for each function call in the underlying engine. It traps all errors and provides error management and administration to prevent accidental GPFs or engine crashes.

Think of it as the "condom layer." While providing the most direct access feasible to the underlying engine and all its capabilities, level 1 of the C++ class **112** also protects the user from the engine. It manages all engine loading and unloading, prevents multiple copies of an engine and calls engines automatically as needed.

The architecture also provides three levels of access: 1. Use the default engine settings. Benefit: No learning up front. Program knowing nothing other than "OCR gets text out of there."2. Prepackage customized engine settings. Set it once for everyone who uses the program, every time they use the program. 3. Modify engine settings at run-time. Let the user choose the settings.

Level 2 of the C++ classes **114** bridges the low-level API calls so they can be used by level 3 **116** in standardized calls for each category. And it supplements the engine by providing additional functionality, such as safely loading and unloading engines.

Level 3 of the C++ class **116** consists of a standardized set of calls for all engines in each category. Programmers can access all the unique functions of each engine in a uniform way.

Another associated C++ class, called a Visual Class **104**, adds a visual interpretation of each engine. This class manages all user interaction with each underlying engine. Like their lower-level counterparts, the Visual Class consists of three layers:

Level 1—**118** adds any dialogs or other pop-up window capability that may be lacking in each engine. Examples: Dialogs to customize the engine settings or, for a recognition engine, the zone definition settings.

Level 2—**120** serves two functions: It bridges level 1 dialogs with the actual Windows window that represents the control. It also handles all Windows-related error message presentation.

Level 3—**122** manages anything else from the underlying engine (such as annotations) that needs to appear on the window. The Visual Class includes engine-specific Windows dialog boxes that let you customize which engine features you want to use, as well as any other Windows representation necessary for a toolkit. (For example, a compression engine has to display the image—the visual class, not the engine, does the work.)

The Object Manager layer **106**, the first horizontal umbrella, orchestrates the underlying objects. It translates service requests into a form that the engine objects can understand.

The Windows Manager **108** presents Windows messages (move window, mouse/scrollbar/toolbox activity) to the Object Manager. It is written using Microsoft's Foundation Class (MFC), which makes it easy to support OCXs. (The OCX is in fact an MFC class.)

At the top, a visual interface **110** presents to the user a set of visual calls and translates those calls into Windows messages. This layer comprises only 5% of the VBX code, yet it permits the toolkit to appear as a VBX, OCX or other standard visual interface.

Accordingly, the present invention provides two main layers, the engine object component layer and the object manager layer. By creating these two main layers, the present invention allows third parties to create their own engine object component layers so that the third party engine can be readily compatible and useable by the present invention. In addition, the present invention is accessible via the Internet. That is, the present invention is operable over the Internet using, for example, standard Internet protocols, such as component object module (COM) communication protocol and distributed COM (DCOM) protocol.

In addition, the present invention optionally combines three layers of functions including the visual interface, the windows manager and the object manager into one layer called the object manager. Of course, this combination of layers is not meant to convey that only these specific layers must be used, but rather, to be indicative of overall functionality generally required to implement or execute component engines. That is, one or more of the above functions may be incorporated into the object manager layer. The present invention also advantageously combines the visual classes and C++ classes into the engine object component to further standardize and/or provide access to the object manager for engine object components.

The present invention optionally uses the standard ActiveX component control supplied, for example, by MicroSoft Corporation. ActiveX is a protocol for component communication. The present invention also creates each of the object manager and the engine component layer as a separate ActiveX. That is, the object manager is its own ActiveX control, and the engine object is its own ActiveX control. Thus, the engine object can now run independently from the object manager. Accordingly, the engine object can operate without relying necessarily on the concurrent operation of the object manager.

The independent relationship between the engine object and the object manager means also that the engine object represents a discrete means of technology. For example, an engine object can be an OCR technology. This provides several benefits. First, because the object manager layer is open, the manufacturer of the OCR technology can wrap their own engine in the form of an engine object component, and the engine will automatically "plug into" or work with, the object manager. Thus, the engine object is provided high level access, making it accessible to many more parties, users, and the like. When the object manager interface is designed to be open, any third party, such as an engine manufacturer, can create their own engine object component that is compatible with the object manager, the manufacturer can do it.

FIG. **19** is an illustration of a distributed environment or architecture for manually and/or automatically generating and/or using reusable software components for client server and/or intranet operating environments. A very significant point that is relevant to why the object manager and the engine object component are independent in the present invention relates to providing a distributed environment for using the present invention. Rather than communicate within the same technology between the object manager and the engine object, the object manager and the engine object component communicate with each other in binary mode, via, for example, standard distributed component object module (DCOM) communication. As illustrated in FIG. **19**, object manager **14** communicates with engine object component **16**, **18**, **20** via DCOM specification **166**. Other types of component communication may also be utilized that provide the capability of a distributed component interaction.

Thus, the engine object component and the object manager can leverage current protocols to not only communicate on the same machine, but also on different machines such as a client server and/or intranet and/or Internet environment. The object manager can be placed on one machine, and the engine object component on another machine and have distributed processing, what is otherwise called thin client processing, distributed processing, wide area intranet processing.

What this allows the present invention to do is to put the object manager on the thin client, who would accept the request from the user, for example, to OCR something or to print something. The actual request is handled or processed by the engine object component which generally resides on the server. The engine object component contains the horse power, or the processing power to process the request.

The engine object layer is generally located in the same or substantially same location as where the core technology or engine itself is being stored. Alternatively, the engine object layer and the engine may be optionally located in a distributed environment on different machines, servers, and the like.

FIG. **20** is a detailed illustration of the distributed environment or architecture for manually and/or automatically generating and/or using reusable software components for client server and/or intranet operating environments. In FIG. **20**, client **170** includes object manager layer **172**. Client **170** executes the core technology **180**, via accessing engine object layer **178** managed/stored on server **176**, and communicated via server **174**.

Client **182**, located on the same server **176** as core technology **180** and engine object layer **178**, may also be used to execute the core technology **180** via object manager layer **184**. In this instance, the client **182** with the object manager layer **184** is located on the same server **176** as the engine object layer **178**. In addition, since the present invention utilizes a communication protocol between components, for example, DCOM, that allows a client to also include both the engine object component layer and the object manager layer on the same machine **186**, as well as the core technology.

Further, since the object manager is formatted or constructed of a client technology, the object manager can sit in a standard browser. This means that anyone that has an Internet browser, i.e., anyone that has access to the world wide web (WEB) can actually access the core engine technology. Thus, by structuring the architecture of the present invention as described herein, users automatically become Internet, intranet and/or WEB enabled.

The present invention also transforms the core technology from essentially client based technology into a client server and/or a thin client technology. This makes the core technology high level accessible, thereby transforming any core technology into client server, or hidden client technology. The browser is located on the client, and the browser leverages the object manager. Accordingly, the browser optionally contains the object manager, and the object manager makes requests over, for example, the Internet, local network, and the like via a server, to the engine object. The server would be either a web server or a LAN server.

The present invention also advantageously provides the ability to have the client and the server, in a distributed environment as discussed above, or on the same machine locally. The present invention utilizes the DCOM communication protocol defining the communication protocol between the object manager and the engine object component. Accordingly, since DCOM can work on the same machine as well as in a distributed environment, DCOM does not necessitate that the engine object or the object manager component be on two separate machines.

FIG. **21** is an illustration of a distributed environment or architecture for manually and/or automatically generating and/or using reusable software components for network environments, such as the Internet. As illustrated in FIG. **21**, object manager **14** communicates with engine object component **16**, **18**, **20** via DCOM specification and a networking environment, such as the Internet, intranet, and the like **168**. Other types of component communication may also be utilized that provide the capability of a distributed component interaction over a networking environment.

FIG. **22** is a detailed illustration of the distributed environment or architecture for manually and/or automatically generating and/or using reusable software components in the Internet environment. In FIG. **22**, client **170** includes object manager layer **172**. Browser/thin client **170***a*executes the core technology **180**, via accessing engine object layer **178** managed/stored on web server **176***a*, and communicated via the Internet **174***a*.

Browser/thin client **182***a*, located on the same web server **176***a*as core technology **180** and engine object layer **178**, may also be used to execute the core technology **180** via object manager layer **184**. In this instance, the browser/thin client **182***a*with the object manager layer **184** is located on the same web server **176***a*as the engine object layer **178**. In addition, since the present invention utilizes a communication protocol between components, for example, DCOM, that allows a client to also include both the engine object component layer and the object manager layer on the same machine **186**, as well as the core technology.

FIGS. **23A–23C** are illustrations of the image viewer user selectable or configurable or programmable interface and/or functionality associated therewith in accordance with the present invention. In FIG. **23A**, user interface **200** for image viewing includes viewing frame **202**, with dual viewing areas **204**, **206**. Viewing area **204** includes at the periphery, previous page activator **208**, at the top, document tools **210**, and at the bottom status indicator **214**. Viewing area **206** includes at the periphery, next page activator **212**, at the top, document tools **214**, and at the bottom status indicator **216**.

Advantageously, this user interface is selectable and/or customizable by the user, as illustrated below in connection with this figure and FIGS. **23B–23C**. Significantly, the image viewer provides the ability to a user to retain or develop a specific perspective on viewing a document. One of the features of the viewer is therefore the ability to change the user's perspective. For example, the user might be looking at the same document, as a book, as a film, or as a bounded or traditional book. This gives the user the ability to relate to the document in a fashion that they are comfortable with, depending on the content or depending on the user. That is, the image viewer is like a usable selectable perspective on viewing a document in a plurality of ways.

FIG. **23B** is an illustration of another user selectable interface for image viewing. In FIG. **23B**, user interface **200'** for image viewing includes viewing frame **202'**, with single viewing area **204'**. Viewing area **204'** includes at the top left, previous page activator **208'** and at the top right next page indicator **212'**. Viewing area **204'** also includes at the left area document tools **210'**, and at the bottom status indicator **214'**. Viewing area **204'** also includes at the top, multiple viewing page area **218**, that appears and preferably moves like a film, and provides viewing of multiple consecutive or non-consecutive pages. Advantageously, this user interface is selectable and/or customizable by the user, as illustrated below in connection with this figure and FIG. **23A** and FIG. **23C**.

FIG. **23C** is an illustration of another user selectable interface for image viewing. In FIG. **23C**, user interface **200"** for image viewing includes viewing frame **202"**, with single viewing area **204"**. Viewing area **204"** includes at the top right, previous page activator **208"** and at the bottom left next page indicator **212"**. Viewing area **204"** also includes at the left area document tools **210"**, and at the bottom status indicator **214"**. Viewing area thus provides a user interface to view a document that appears like a bound or more traditional book. Advantageously, this user interface is selectable and/or customizable by the user, as illustrated below in connection with this figure and FIGS. **23A–23B**.

FIG. **24** is an illustration of a stand-alone and/or distributed environment or architecture for image viewer in client server and/or intranet operating environments. The architecture in FIG. **24** provides the capability to perform the viewer process off-line. That is, the viewer process **188** provides an added feature on top of the object manager layer **14**. As described above, object manager layer **14** is essentially an interface, and the viewer process **188** is an application that leverages the object manager layer **14**.

The advantage of the viewer process **188** being built on the object manager layer **14**, which is built on top of the engine object layer **16**, **18**, **20**, is that the viewer process can offset its processing capabilities anywhere in a distributed environment. It can have the processing occur at the local station, on a server, and the like, as described below in detail. Significantly, the object manager and the engine object component are independent to provide a distributed environment for using the present invention. Rather than communicate within the same technology between the object manager and the engine object, the object manager and the engine object component communicate with each other in binary mode, via, for example, standard distributed component object module (DCOM) communication.

As illustrated in FIG. **24**, object manager **14** communicates with engine object component **16**, **18**, **20** via DCOM specification **166**. Other types of component communication may also be utilized that provide the capability of a distributed component interaction. Object manager **14** is also respectively connectable to viewer process **188**.

Thus, the engine object component and the object manager can leverage current protocols to not only communicate on the same machine, but also on different machines such as a client server and/or intranet and/or Internet environment. The object manager and/or viewer process can be placed on one machine, and the engine object component on another machine and have distributed processing, what is otherwise called thin client processing, distributed processing, wide area intranet processing.

What this allows the present invention to do is to put the object manager on the thin client, who would accept the request from the user, for example, to perform the viewer process. The actual request is handled or processed by the engine object component which generally resides on the server. The engine object component contains the horse power, or the processing power to process the request.

The engine object layer is generally located in the same or substantially same location as where the core technology or engine itself is being stored. Alternatively, the engine object layer and the engine may be optionally located in a distributed environment on different machines, servers, and the like.

FIG. **25** is a detailed illustration of a stand-alone and/or distributed environment or architecture for image viewer in client server and/or intranet operating environments. In FIG. **25**, client **170** includes object manager layer **172** with viewer process **192**. Client **170** executes the core technology **180**, via accessing engine object layer **178** managed/stored on server **176**, and communicated via server **174**. Viewer process **190** is also optionally available to either or both servers **174**, **176**.

Client **182**, located on the same server **176** as core technology **180** and engine object layer **178**, may also be used to execute the core technology **180** and/or viewer process **192** via object manager layer **184**. In this instance, the client **182** with the object manager layer **184** is located on the same server **176** as the engine object layer **178**. In addition, since the present invention utilizes a communication protocol between components, for example, DCOM, that allows a client to also include both the engine object component layer, viewer process **194** and the object manager layer on the same machine **186**, as well as the core technology.

Further, since the object manager is formatted or constructed of a client technology, the object manager can sit in a standard browser. This means that anyone that has an Internet browser, i.e., anyone that has access to the world wide web (WEB) can actually access the core engine technology and/or viewer process. Thus, by structuring the architecture of the present invention as described herein, users automatically become Internet, intranet and/or WEB enabled.

The present invention also transforms the core technology and/or viewer process from essentially client based technology into a client server and/or a thin client technology. This makes the core technology high level and/or viewer process accessible, thereby transforming any core technology and/or viewer process into client server, or hidden client technology. The browser is located on the client, and the browser leverages the object manager. Accordingly, the browser optionally contains the object manager, and the object manager makes requests over, for example, the Internet, local network, and the like via a server, to the engine object. The server would be either a web server or a LAN server.

The present invention also advantageously provides the ability to have the client and the server, in a distributed environment as discussed above, or on the same machine locally. The present invention utilizes the DCOM communication protocol defining the communication protocol between the object manager and the engine object component. Accordingly, since DCOM can work on the same machine as well as in a distributed environment, DCOM does not necessitate that the engine object or the object manager component be on two separate machines.

FIG. **26** is an illustration of a stand-alone and/or distributed environment or architecture for image viewer in network environments, such as the Internet. As illustrated in FIG. **21**, object manager **14** communicates with engine object component **16**, **18**, **20** via DCOM specification and a networking environment, such as the Internet, intranet, and the like **168**. In addition, object manager layer **14** also advantageously communications with viewer process **188***a*. Other types of component communication may also be utilized that provide the capability of a distributed component interaction over a networking environment.

FIG. **27** is a detailed illustration of a stand-alone and/or distributed environment or architecture for image viewer in the Internet environment. In FIG. **27**, client **170** includes object manager layer **172**. Browser/thin client **170***a* executes the core technology **180** and/or viewer process **192***a*, via accessing engine object layer **178** managed/stored on web server **176***a*, and communicated via the Internet **174***a*. Viewer process **190** is also optionally available to web server **176***a*.

Browser/thin client **182***a*, located on the same web server **176***a* as core technology **180**, viewer process **192***a* and engine object layer **178**, may also be used to execute the core technology **180** via object manager layer **184**. In this instance, the browser/thin client **182***a* with the object manager layer **184** is located on the same web server **176***a* as the engine object layer **178**. In addition, since the present invention utilizes a communication protocol between components, for example, DCOM, that allows a client to also include both the engine object component layer and the object manager layer on the same machine **186**, as well as the core technology and viewer process.

The purpose of the Virtual Copier ("VC") aspect of the present invention is to enable a typical PC user to add electronic paper processing to their existing business process. VC is an extension of the concept we understand as copying. In its simplest form it extends the notion of copying from a process that involves paper going through a conventional copier device, to a process that involves paper being scanned from a device at one location and copied to a device at another location. In its more sophisticated form, VC can copy paper from a device at one location directly into a business application residing on a network or on the Internet, or visa versa. The VC invention is software that manages paper so that it can be electronically and seamlessly copied in and out of devices and business applications (such as Microsoft Office, Microsoft Exchange, Lotus Notes) with an optional single-step Go operation. The VC software can reside on a PC, LAN/WAN server, digital device (such as a digital copier), or on a web server to be accessed over the Internet.

Virtual Copier is designed to solve the corporate paper problem by enabling existing web-based and client-server applications to manage paper as part of their solution. Virtual Copier links the familiar and universal world of paper and digital devices to web-based and client-server applications. The result is that the automated business processes become the primary storage of paper in electronic form. Information that is typically managed and processed in paper form is "copied" into the system and managed by the business processes with which users are accustomed, which is made possible by using Virtual Copier. Simple extensions of Virtual Copier support seamless electronic outsourcing of paper processing and archival services over the web.

Virtual Copier is a unique combination of an intuitive application built on an open component architecture that delivers a simple innovation: provide paper processing to existing Intranet and client-server business processes without any fuss. Whether it is an office clerk that needs to easily copy a report from a desktop scanner to the company's Intranet-networked copier, or an accounting software integrator that wants to embed paper processing, Virtual Copier offers a simple solution. To the office clerk Virtual Copier is a document imaging application packaged in the familiar setting of an office copier. To the integrator, the underlying open architecture of Virtual Copier offers a simple integration path for embedding paper processing into its client-server or web-based software solution.

Although managing paper manually is one of the great problems facing corporations, there has been little innovation in enabling those workers to eliminate the need to continuously work with paper manually. Much of the problem stems from the complexity of traditional document management systems, which require days of training and months to become familiar with the system in order to be proficient. Virtual Copier was designed to be as simple as a copier to operate, and yet still provide the complete capability of integrating paper with existing business applications. By simplifying the interface and underlying software infrastructure, VC can manage paper in electronic form as easily as is currently done in physical form.

VC extends the notion of a copier, which simply replicates the image of an original document onto another piece of paper using a single GO or START button, to do a similar operation in software so that the image gets seamlessly replicated into other devices or applications or the Internet.

An example of this is the actual implementation of Virtual Copier as a consumer product. As shown in FIGS. **28** and **29**, the interface of the consumer product called Virtual Copier has a Go button much like a physical copier. This GO button can copy paper, whether physical or electronic, from one device and or application to another device and/or application.

What makes Virtual Copier as simple as its physical counterpart in at least one embodiment is the fact that it replicates the identical motions that a user who is making a copy using a physical photocopier goes through. When a user photocopies a document, he/she selects where they want to copy from (i.e. the sheet feeder), where the user wants to copy to (i.e. 6 copies collated and stapled) and then presses a GO button to actually carry out the photocopy process. With Virtual Copier the process feels familiar because the sequence is the same as illustrated in FIG. **30** with just the Power VC portion of the main Virtual Copier window.

The power of Virtual Copier is the fact that the From can be a physical device (e.g. digital copier, fax or scanner) or an application (e.g. Lotus Notes, Microsoft Exchange, the Internet, or an electronic filing system). The To can also be a physical device (e.g. a fax, digital copier, or printer) or an application (e.g. Lotus Notes, Microsoft Exchange, the Internet, or an electronic filing system). Even though paper is being copied electronically from devices to applications, from applications to devices, from devices to devices, or from applications to applications, the user simply has one sequence to execute: select From, select To, and then press

GO. Virtual Copier will accomplish all translations between device and applications automatically and seamlessly.

Another reason that paper is still a major corporate issue is that traditional document management systems require that a company invest in a whole new system just to store electronic images. Although this is the only way that document management systems have been designed and delivered, it is in fact highly inefficient. Most companies already manage information about physical documents in some form of software applications.

For example, accounting systems have long been used to maintain information about invoices and bills that arrive into a company from outside sources as physical pieces of paper. When an invoice arrives, its information is keyed into the accounting software, where balances are maintained and accounts payable information is coordinated. Yet the original invoice is stored manually, and every time that a request is made for a copy of the signed invoice, someone manually retrieves the invoice from a physical filing cabinet. Accounting systems, like most business applications, typically have no way of maintaining an electronic copy of the physical invoice, and adding a document management system to an accounting system is cumbersome, costly, and difficult to maintain, and even more difficult to coordinate.

Virtual Copier solves this problem in at least one embodiment by copying paper directly into the existing accounting system. Simply adding a To item in the Virtual Copier window enables a user to copy paper directly into the appropriate accounting record of the existing accounting system. This requires no retraining (users who are trained on the accounting system will still use the accounting system in the same way), requires no document management system (the electronic copy of the document is actually being maintained by the accounting system itself), there is no coordination between two systems (Virtual Copier embeds the invoice with the appropriate accounting record), and it is simple (one Go button).

What is true with regard to the example above of an accounting system is true of most other business applications. The power of Virtual Copier is that it can turn an information system into a document management system by adding support for electronic paper directly into the existing business application, whether it is a client, server-based, or web-based system.

Virtual Copier enables corporations to perform sophisticated document imaging with their existing Web-based and client-server applications through a user interface that is as familiar as the office copier. Virtual Copier can be used out-of-the-box as a standalone application to copy, scan, fax, or print images using existing digital devices within corporate environments or across the web. With the extensions, as described below, Virtual Copier can be integrated into Web-based and client server applications, such as ERP or accounting systems, to eliminate paper from existing business processes and legacy applications. Virtual Copier can also be used to support seamless access to document image processing and archival over the web since, in at least one embodiment, the VC interface is implemented as a software application.

VC is architected as an application that delivers end-user functionality while remaining open to third-parties extensions. For example, VC can be viewed as a copier. Like a copier, VC takes paper in, and produces paper going out. The only difference is that VC does not distinguish between electronic and physical paper.

To accommodate third-party extensions, VC is divided into five essential modules. Each module is a counterpart to an aspect that is found on a conventional copier. Based on the modular design of VC, each aspect of VC can be independently extended, offering much greater flexibility than conventional copiers.

The five core modules of VC illustrated in FIG. **31** are:

Input Module—The Input Module manages paper or electronic paper entering VC. This module manages imaging devices to input paper through scanners, MFPs, or the new breed of digital copiers. The Input Module also manages reading electronic paper from third-party or proprietary applications. The counterpart to VC's Input Module on a conventional copier is the scanner subsystem.

Output Module—The Output Module manages paper or electronic paper exiting VC. Like the Input Module, this module manages imaging devices to output paper to standard Windows printers, specialty image printers, MFPs, or the new breed of digital copiers. The Output Module also manages writing electronic paper to third-party or proprietary applications. The counterpart to VC's Output Module on a conventional copier is the printer or fax subsystem.

Process Module—The Process Module applies processing to the electronic paper as it is being copied. Examples of a process are OCR and ICR. The Process Module can also apply non-imaging functionality as well, such as workflow or other relevant tie-ins to the electronic paper as it is being copied. One of the advantages of VC over conventional copiers is that multiple processes can be applied to a single virtual copy. The counterpart to VC's Process Module on a conventional copier is the controller.

Client Module—The Client Module presents the electronic paper as it is being copied, and any relevant information related to the input or output functions. For example, if the Output Module is directed to a printer, then the Client Module might present the finishing capabilities; if the Output Module is directed to Goldmine, then the Client Module might present the target contact record to which the document is being copied. The counterpart to VC's Client Module on a conventional copier is the panel.

Server Module—Unlike conventional copiers, VC's Server Module is a unique subsystem that can communicate with the other modules as well as third-party applications. The Server Module is what makes VC a far more powerful concept than simply an application that can control a scanner and a printer to mimic a copier. The Server Module can be used to combine third-party applications with the new breed of digital imaging devices to create unique and custom virtual copier solutions. A virtual copier can be created with VC by combining a scanner with a printer; or by combining a scanner with an application; or by combing an application with an image printer. In each case VC is dynamically creating a custom virtual copier, with a complete understanding of how paper flows from the source to its destination. There is no counterpart to VC's Server Module on a conventional copier.

One of the primary design goals of VC is to make it simple to integrate VC with third-party applications. There are two options to integrating VC into a third-party application: running VC as an external service, or embedding VC as an underlying service.

VC is in one embodiment and optionally a standalone application that enables a user to scan (copy) paper from a device to a third-party application, and to print (copy) the reference of an image document from a third-party application to a printing device. VC does not require the third-party application to be aware that VC is operating. Rather, VC recognizes that the third-party application is running, and it intelligently copies paper to and from that application as illustrated in FIG. 32.

In this scenario the user is interacting with VC's Client Module in order to execute a copy operation to and from the third-party application. There does not have to be any changes made to the third-party application, not even to its interface, in order for VC to operate. The user of VC only knows that to copy to and from the third-party application, a custom Input and Output Module must be selected, and the Go button is pressed.

In order to support copying to and from a third-party application, VC must be able to support extensions that understand each third-party application. This is accomplished through the Input and Output Modules. The Client, Server, and even Process Modules remain independent across third-party applications. However, in order to support outputting to a third-party application, an Output Module is developed that is unique to that third-party application. Likewise, an Input Module is developed that is unique to a third-party application in order to support reading images from that application.

It is the optional Input and Output Modules that render VC extendable. For each third-party application there is a unique pair of Input and Output Modules that understand the third-party application, and how to copy images to and from that application. Each Input and Output Module registers itself to the Windows registry so that the Server Module knows how to find them. In this way Virtual Copier can grow indefinitely, to support any number of third-party applications.

The significant point is that the Input and Output Modules have their own interface, and can be developed independently from any other module. As long as the Input and Output Module conform to the API specified in this document it will plug-and-play with VC. VC will be able to mix and match the custom Input and Output Module with its standard and other custom Input and Output Modules.

A third-party application can also use the services of VC without its user interface. That is, a third-party application can embed VC's functionality and provide its own interface to its functionality. For example, rather than have VC as a separate application, a special button can be placed on a third-party application that launches VC in the background as illustrated in FIG. 33.

VC is designed so that the Server Module can run independently from the Client Module. All the core functionality, including communicating with the Input, Output, and Process Modules, are performed directly by the Server Module. The Client Module is generally simply an interface to the Server Module. Therefore, all the services of the Server Module can be made available in the background to a third-party application without the need for an interface. The third-party application can in fact become the user's interface to VC.

In order to support VC operating in the background a third-party application merely has to communicate with the Server Module directly, as described later in this document. The Server Module, as all modules in VC, support COM-based interfaces for simple and direct support from all major Windows development environments.

At the heart of VC is the Server Module. A virtual copy operation can only be initiated using the Server Module. The Server Module coordinates the activities of the various modules while maintaining the information regarding the current process and document. It also collects and passes information from one module to another regarding the document and process. Events and an API are used to control the modules and their interaction with each other as well as with the Server Module.

The following are the main functions of the Server Module:

Enable Virtual Copy Operation—The Server Module provides simple methods to initiate, cancel, and reset VC. The API is designed to imitate the simplicity of using a conventional copier.

Maintain List of Available Modules—The Windows registry contains the list of available Input, Output, and Process Modules that can be used with VC. The Server Modules reads this list on startup, and maintains it in the Modules object that can be accessed by the other modules. Although each module can read this information directly from the registry, it is preferable to use the Modules object. All information regarding the available modules can be found in the Modules object.

Maintain the Currently Active Modules—The Server Module maintains the current Input, Output, and Process Modules that are being used for the current virtual copy operation. This is maintained in the Program object. This information can also be saved to disk in a Process Template file.

Maintain Complete Document Information—The Server Module maintains all the information regarding the current file being copied. This is maintained in the VDocument object. This information can also be saved to disk in a Document Template file.

As with other design elements of VC, the VC logic flow illustrated in FIG. 34 parallels the basic logic flow of a conventional copier. In a conventional copier, paper is pulled into the copier, processed, and output. Likewise, in VC the Server Module initiates the Input Module, Process Module, and Output Module in that sequence. Unlike a conventional copier which does not have the ability to update its panel, VC updates its Client Module as well as the results of each Module acting on the document as illustrated in FIG. 35.

All actions to create, process, and write images are the responsibility of the Input, Process, and Output Modules respectively. The Server Module is a scheduler of activities, providing the information and initiating the modules at the appropriate time in the virtual copy operation. The Server Module manages the other Modules. It does not know about the internal workings of the modules, nor the contents of the information being copied. The Server Module API is sufficiently rich to maintain all the information necessary for a basic virtual copy operation.

The Server Module API is divided, for example, into the following COM-based interfaces:

Modules Object—This object maintains the list of available Input, Output, and Process Modules

Program Object—This object maintains the currently selected Input, Output, and Process Modules

VDocument Object—This object maintains the information regarding the current document that is being copied

VC Methods—These methods are used to initiate, cancel, and reset VC

VC Events—These events are used to provide feedback to the Client Module

The purpose of the Modules object is to provide the Client Module with the full list of available Input, Output, and

Process Modules that is available to the user. The Client Module can obtain the user-readable names for each module, as well as its icon and other key information. The Modules object is primarily used to seed list or combo boxes that provide the end-user with a choice of modules from which to select.

In a preferred embodiment, the Modules Object has, for example, the following structure illustrated in FIG. **36**, however, alternative structures and/or functionality may optionally be used for this object and/or other objects used in the present invention:

Name	Configure
Type	Method
Format	.Configure( )
Description	The Configure method causes the module to prompt the user for configuration information. Each module maintains its own configuration dialog, and therefore may look different than other modules.
Sample	VCopier.InputModules(1).Configure( )
Name	Default
Type	Property; Object of type InputModule, OutputModule, or ProcessModule
Format	.Default - Read Only
Description	The Default property identifies the default module that the Server Module will use at startup or if no other module is identified.
Sample	MyInputModule = VCopier.InputModules.Default
Name	ID
Type	Property; BSTR
Format	.ID- Read Only
Description	The ID property identifies the ProgID of a module. The ProgID can be used to derive other information about the module, including its Icon.
Sample	ModuleName = VCopier.InputModules(1).ID
Name	File
Type	Property; BSTR
Format	.File-Read Only
Description	The File property identifies the full pathname of the physical file of a module.
Sample	FileName = VCopier.InputModules(1).File
Name	InputModule, OutputModule, ProcessModule
Type	Object
Format	.InputModule, .OutputModule, .ProcessModule-Read Only
Description	The InputModule, OutputModule and ProcessModule are the individual objects maintains by the InputModules, OutputModules, and ProcessModules collections respectively. Each one of these objects has the following elements: Name ID File Configure The Name property is BSTR that is the user-readable name of the module. The ID is a BSTR that represents the ProgId of the module. The File property is a BSTR that is the full pathname of the module. The Configure method prompts the user with a dialog for configuring that module.
Sample	MyInputModule = VCopier.InputModules(2)
Name	InputModules, OutputModules, ProcessModules
Type	Collection of InputModule, OutputModule, and ProcessModule objects respectively
Format	.InputModules, .OutputModules, .ProcessModules-Read Only
Description	The InputModules, OutputModules, and ProcessModules collections maintain the list of available modules for each category. Each collection maintain the following information: InputModule/OutputModule/ProcessModule Default

-continued

	The first element is the individual module in the collection of modules that are available to VC. The Default object is the default module that VC uses at startup. The Server Module maintains these collections under the Modules object.
Sample	MyInputModule = VCopier.InputModules(2)
Name	IsLoaded
Type	Method, Boolean
Format	.IsLoaded( )
Description	The IsLoaded method returns True if the module is loaded into memory, and False if it is not.
Sample	ModuleName = VCopier.InputModules(1).IsLoaded
Name	Load
Type	Method
Format	.Load( )
Description	The Load method manually loads the module into memory. Once a module is loaded in VC it remains in memory until it is specifically unloaded using the Unload method, or the program exits.
Sample	ModuleName = VCopier.InputModules(1).Load
Name	Name
Type	Property, BSTR
Format	.Name-Read Only
Description	The Name property identifies the user-readable name of a module. This name can be used in a list box for a user to select the module.
Sample	ModuleName = VCopier.InputModules(1).Name
Name	ResetSettings
Type	Method
Format	.ResetSettings( )
Description	The ResetSettings method returns the settings of the module back to its original state when the VC first called it. A user can change the settings of a module when it is configured. This method is used to role back changes made bu a user during the VC session. To save the settings between sessions, use the SaveSettingsAsDefault method.
Sample	ModuleName = VCopier.InputModules(1). ResetSettings( )
Name	SaveSettingsAsDefault
Type	Method
Format	.SaveSettingsAsDefault( )
Description	The SaveSettingsAsDefault method save any changes to the settingst the user has done during the session to disk so that they become the new settinsg.
Sample	ModuleName = VCopier.InputModules(1) ResetSettings( )
Name	Unload
Type	Method
Format	.UnLoad( )
Description	The Unload method manually unloads the module from memory. Once a module is loaded in VC it remains in memory until it is specifically unloaded using the Unload method, or the program exits.
Sample	ModuleName = VCopier.InputModules(1).Unload( )

The Program Object maintains the currently selected Input, Output, and Process Modules. It is generally set by the Client Module based on input from a user. However, in applications that do not have a user interface the program object can be used to directly set the modules to run VC. The Program Object has the following structure illustrated in FIG. **37**.

All elements of the Program Object are defined in the Modules Object section. The VDocument Object maintains information about the current document being copied. The VDocument represents a virtual document rather than a physical one. It is designed to allow the flexible management of multi-image files that together constitute an document. The internal VDocument maps to physical files as illustrated in FIG. **38**.

The VDocument Object calculates the total number of pages of all the files associated with it, and lays out each page of each document in a single virtual document. As the figure illustrates, if 4 files contain a total of 8 pages, then VDocument considers this an 8 page document. If the 6th page is requested, VDocument will return the second page of File C in the above figure. This enables VDocument to handle single page files that together constitute a document (as is the case with many of the new digital copiers), or a single multi-page image file, or any combination of the two. The VDocument Object is illustrated in FIG. 39 and below.

Name	Add
Type	Method
Format	.Add(BSTR File, Long Page)
Description	The Add method is used to add a page to the VPages collection. The two arguments File and Page represent the disk file and the page number to associate with the new page in VPages. One page of one file is added at a time using this method.
Sample	VDocument.Add(FileA, 2)
Name	AutoDelete
Type	Property, Boolean
Format	.AutoDelete
Description	The AutoDelete property lets the Server Module know whether to delete the files once the virtual copy operation is completed. When set to True the Server Module will delete the physical disk files maintained in Vdocument either before the next virtual copy operation, or when VC is shut down. When set to False Vdocument is cleared of its contents between virtual copy operations, but the actual files are not deleted from the disk. In general if the VDocument object points to existing files then AutoDelete should be set to False. If the VDocument object points to temporary files, then AutoDelete should be set to True so that the disk files are cleaned up (i.e. deleted) by the Server Module. By default AutoDelete is set to False.
Sample	VCopier.VDocument.AutoDelete = True
Name	Clear
Type	Method
Format	.Clear( )
Description	The Clear method is used to empty the contents of the Vdocument object. The VPages object is emptied and the reference to files are deleted in conformance with the AutoDelete property.
Sample	VDocument.Clear( )
Name	File
Type	Property, BSTR
Format	.File
Description	The File property of the VPage object points to the disk file that contains the image associated with the VPage page.
Sample	MyFile = VDocument.VPages(2).File
Name	Page
Type	Property, Long
Format	.Page
Description	The Page property of the VPage object points to the image offset into the disk file that contains the image associated with the VPage page.
Sample	MyPage = VDocument.VPages(2).Page
Name	Remove
Type	Method
Format	.Remove( )
Description	The Remove method is used to remove a page from the VPages collection. The single argument Index is the offset page into the VPages collection.
Sample	VDocument.VPages(2).Remove( )
Name	Vpage
Type	Object
Format	.Vpage
Description	Each VPage object represents a single virtual page in the VDocument object. Each VPage object contains the name of the file that contains its virtual page in the .File propety, and a .Page property which is the page offset in the image file.
Sample	MyPage = VCopier.VPages(2)
Name	Vpages
Type	Collection of Page objects
Format	.Vpages
Description	The VPages collection contains one VPage object per virtual page. Each page of each image file that is tracked by VDocument is considered a unique page, and its information is maintained by a VPage object.
Sample	MyPage = VCopier.VPages(2)

-continued

The Server Module supports simple methods that accomplish the basic copier functionality of go, cancel, and reset. The Server Modules has the following structure:

Name	Cancel
Type	Method
Format	.Cancel( )
Description	The Cancel method is used to cancel the currently running virtual copy operation. The Cancel method can only be used once the Go method is called and prior to its completion.
Sample	VCopier.Cancel( )
Name	Go
Type	Method
Format	.Go( )
Description	The Go method is used to initiate a virtual copy operation. It calls the modules in the following sequence: Program.InputModule, Program.ProcessModules, and then Program.OutputModule. The virtual copy operation can be cancelled prior to its completion by calling the Cancel method.
Sample	VCopier.Go( )
Name	Reset
Type	Method
Format	.Reset( )
Description	The Reset method is used to clear the contents of the Program object. After calling the Reset method VC is considered to have no assigned Input and Output modules selected. The modules that are reset are not unloaded from memory.
Sample	VCopier.Reset( )

The are two events that the Server Module supports: Error and Status. The Error event is generated anytime any of the Modules produce an error condition. The Status event is generated when information needs to be transferred between the IOP or Server Modules and the Client Module.

The following are details for each event, illustrated in FIGS. 40 and 41 and below.

Name	Error
Type	Event
Format	.Error(. . .)
Description	The Error event is triggered whenever there is an error by one of the modules. The error can be trapped and displayed or processed by the Client Module.
Sample	
Name	ErrorCode
Type	Argument, Long
Format	.ErrorCode
Description	The ErrorCode argument of the Error event identifies the actual error code. There are no predefined error codes for all modules. Each module produces its own set of error codes.
Sample	
Name	ErrorText
Type	Argument, BSTR
Format	.ErrorText

-continued

Description	The ErrorText argument of the Error event identifies the actual error text. There are no predefined error texts for all modules. Each module produces its own text for its error codes.
Sample	
Name	ModuleID
Type	Argument, BSTR
Format	.ModuleID
Description	The ModuleID argument of the Error event identifies the source of the error condition. The ModuleID is defined as the version-dependent ProgID.
Sample	
Name	Severity
Type	Argument, Long
Format	.Severity
Description	The Severity argument of the Error event identifies the level of error condition. The following levels are currently implemented: 1-Severe 2-Warning
Sample	
Name	SubModuleID
Type	Argument, BSTR
Format	.SubModuleID
Description	The SubModuleID argument of the Error event identifies the secondary source of the error condition. The SubModuleID can defined as the version-dependent ProgID, or any other value determined by the Module that generates the error condition.
Sample	
Name	URL
Type	Argument, BSTR
Format	.URL
Description	The URL argument of the Error event identifies the URL address (web site, HTML file, or resource file URL). that contains the HTML representation of the error condition. The information presented can be more dynamic as well as formatted than the ErrorText argument.
Sample	
Name	Info1, Info2
Type	Argument, Variant
Format	.Info1, .Info2
Description	The Info1 and Info2 arguments of the Status event are placeholders for additional information that needs to be supplied with specific status numbers
Sample	
Name	Status
Type	Evenet
Format	.Status(. . .)
Description	The Status event is trigerred by any of the modules when there is information that needs to be relayed to the user or the Client Module.
Sample	
Name	StatusNumber
Type	Argument, Long
Format	.StatusNumber
Description	The StatusNumber argument of the Status event identifies the actual status code. The values between 1 and 1000 are private and cannot be generated by an IOP Module for private use. Any other status numbers are open for private IOP Module use.
Sample	
Name	StatusText
Type	Argument, BSTR
Format	.StatusText
Description	The StatusText argument of the Status event identifies the actual status text.
Sample	
Name	StatusType
Type	Argument, BSTR
Format	.StatusType
Description	The StatusType argument of the Status event identifies the type of status.
	1 - Informational 2 - Instruction
Sample	

The Server Module broadcasts the Status event to the Client Module. There are standard status events that the Server Module generates which the Client Module can rely on. These are the events that manage the flow of modules and user interaction with the Server Module. The following is a general workflow of the events that are generated is illustrated in FIG. **42**.

StatusNumber	StatusText	Description
	VseModuleCanceled	The IOP Module canceled the operation by setting the Cancel argument in the Feedback.Error or Feedback.Status methods to True.
	VseModuleConfigureEnd	The IOP Module has completed presenting its configuration dialog
	VseModuleConfigureStart	The IOP Module has started presenting its configuration dialog
	VseModuleGoEnd	The IOP Module has ended executing
	VseModuleGoStart	The IOP Module has started executing
	VseModuleLoadEnd	The IOP Module has completed loading
	VseModuleLoadStart	The IOP Module has started loading
	VseModuleUnloadEnd	The IOP Module has completed unloading
	VseModuleUnloadStart	The IOP Module has started unloading
	VseProgramCanceled	The Server Module has canceled executing (using the .Cancel method)
	VseProgramEnd	The Server Module has ended executing
	VseProgramStart	The Server Module has started executing a Go operation

The Client Module presents to the user information regarding the copy process, and initiates the virtual copy through the Server Module. The Client Module can be a GUI that Imagination Software develops, or a third-party application that directly communicates with the Server Module. The goal of the Client Module is to capture sufficient information and pass that information along to the Server Module in order to initiate a single virtual copy.

The Client Module follows the following general logic flow illustrated in FIG. **43**. The first step for the Client Module is to determine that the Server Module exists, and to successfully launch the Server Module. This is done using a standard COM interface.

If the Client Module is a GUI then it can present icons and the names of all the available Input, Output, and Process Modules for the user to select. The Client Module does not need to know any information about these modules. All names and ProdId's are available from the Server Module API using the Modules Object.

If the user selects a new Input or Output module, the Client Module updates the appropriate Program.InputModule, Program.OutputModule, or Program. Process Modules object available on the Server Module.

At any time the Client Module can initiate the Go method of the Server Module. This is a synchronous process—once the Go method is initiated the only way to stop it is to call the Cancel method. Only one Go method can be called at a time, and it must run to completion before another one is called.

During the virtual copy, the Server Module will send back Status and Error events that should be processed and displayed (if there is a GUI) by the Client Module. The only requirement for a Client Module is that it at least substantially conforms to the interface described in the Server Module section. The architecture described in this section, and its associated sample source code, is designed to facilitate development of Client Modules by third parties. It should be used as a guide for developing a Client Module—it is not the only way a Client Module can be designed.

The internal architecture described below is generally independent from the interface requirements for a Client Module. The Client interface must be implemented regardless of whether or not the Client is designed with the architecture described in this section. The basic Client architecture is illustrated in FIG. **44**.

The Input, Process, and Output ("IOP") Modules extend VC by enabling specialized hardware and software to interact with VC. Each IOP Module understands the input, output, or processing capabilities of a specific technology, as well as how to communicate with the Server Module. In this way an IOP Module can read or write images to and from any device or software application while still being managed by the Server Module. To the user of VC, interacting with any device or software application is the same.

The IOP Modules share a common API to facilitate communication with each other, with the Server Module, as well as with third-party programs. The interface is based on COM. Both the Server Module as well as third-party applications can communicate with the Input, Process, and Output Modules using the specified COM interface. Additionally, third-party vendors can create their own versions of the Input, Process, and Output Modules as long as they conform to the specified COM interface.

The following are the main functions of the Input, Process, and Output ("IOP") Modules:

Respond to Server Module Go( ) Method—The Server Module calls the other modules using a COM-based Go( ) method. All necessary information regarding the virtual copy operation is passed along using arguments of the Go( ) method. The IOP module can then handle its internal operation independent of any other module.

Generate Status & Error Feedback—The IOP module should let the Server Module know its progress, error conditions, or any other useful process or userbased information.

Initiate Communication With the Server Module—The IOP Module can at any time initiate communication with the Server Module to provide new information. This enables the IOP Module to pole the device or software application that it is linked to, and convey that information back to the Server Module.

The API for the Input, Process, and Output Modules are deliberately made simple so that third-party vendors can create their own custom versions of these modules with relative ease. The API, illustrated in FIG. **45**, consists of the following COM-based interface:

Go(VDocument, Feedback)—This is the single method that initiates a module to complete its phase of the virtual copy. The Go( ) method is called by the Server Module when it is ready to execute the functionality of the module. The two parameters are the VDocument object, which contains the information about the current document being copied. The module can update the VDocument with additional images, as is typical of an Input Module, or simply read and process the document, as is typical of an Output Module. The second parameter is a Feedback object, which contains the two events that the IOP module can generate back to the Server Module.

Name	Configure
Type	Method
Format	.Configure( )
Description	The Configure method causes the module to prompt the user for configuration information. Each module maintains its own configuration dialog, and therefore may look different than other modules.
Sample	MyInputModule.Configure( )
Name	Go
Type	Method
Format	.Go(VDocument, Feedback)
Description	The Go method is called by the Server Module to initiate the IOP module to execute its part of the virtual copy opemtion. The Vdocument Object is passed along as an argument so that the IOP module can add to or read the current document that is being processed. Refer to the Server Module section for a complete description of the VDocument object. The second parameter of the Go method is a Feedback object. The Feedback object enables the IOP module to send status and error updates back to the Server Module. These events are also described in the Server Module section.
Sample	IOP.Go(VDocument, Feedback)
Name	ResetSettings
Type	Method
Format	.ResetSettings( )
Description	The ResetSettings method returns the settings of the module back to its original state when it was first called. A user can change the settings of a module when it is configured. This method is used to role back changes made by a user during the VC session. To save the settings between sessions, use the SaveSettingsAsDeafult method.
Sample	MyInputModule.ResetSettings( )
Name	SaveSettingsAsDefault
Type	Method
Format	.SaveSettingsAsDefault( )
Description	The SaveSettingsAsDefault method save any changes to the settings the user has done during this session to disk so that they become the new settings.
Sample	MyInputModule.ResetSettings( )

The Feedback object illustrated in FIGS. **46**–**47** is used to communicate between the IOP and the Server Module. The Feedback object supports two methods that are used like events. The purpose of this mechanism is to limit the communication between the IOP and the Server Module to just those objects presented to the IOP Module by the Server Module through the Go method. In this way the IOP Module is handed all the information it needs to execute its part of a copy operation.

The Feedback object contains two methods: Error and Status. The Error event is used to respond back to the Server Module all error conditions. The Status method is used to communicate back to the Server Module all information updates, such as progress.

The following are details for each of these methods:

Name	Cancel
Type	Argument, Boolean Reference
Format	.Cancel
Description	The Cancel argument of the Error method is used to establish whether the Server Module will continue with the virtual copy operation once this IOP is completed. If set to True then the Server Module will not continue its virtual copy operation. The Server Module will wait until the IOP Module returns on its own. The Server Module does not shut down the IOP Module.
Sample	
Name	Error
Type	Method
Format	.Error(. . .)
Description	The Error event is triggered whenever there is an error by one of the modules. The error can be trapped and displayed or processed by the Client Module.
Sample	
Name	ErrorCode
Type	Argument, Long
Format	:ErrorCode
Description	The ErrorCode argument of the Error event identifies the actual error code. There are no predefined error codes for all modules. Each module produces its own set of error codes.
Sample	
Name	ErrorText
Type	Argument, BSTR
Format	.ErrorText
Description	The ErrorText argument of the Error event identifies the actual error text. There are no predefined error texts for all modules. Each module produces its own text for its error codes.
Sample	
Name	Severity
Type	Argument, Long
Format	:Severity
Description	The Severity argument of the Error event identifies the level of error condition. The following levels are currently implemented: 1 - Severe 2 - Warning
Sample	
Name	SubModuleID
Type	Argument, BSTR
Format	:SubModuleID
Description	The SubModuleID argument of the Error event identifies the secondary source of the error condition. The SubModuleID can defined as the version-dependent ProgID, or any other value determined by the Module that generates the error condition.
Sample	
Name	URL
Type	Argument, BSTR
Format	.URL
Description	The URL argument of the Error event identifies the URL address (web site, HTML file, or resource file URL), that contains the HTML representation of the error condition. The information presented can be more dynamic as well as formatted than the ErrorText argument.
Sample	
Name	StatusText
Type	Argument, BSTR
Format	.StatusText
Description	The StatusText argument of the Status event identifies the actual status text.
Sample	
Name	StatusType
Type	Argument, BSTR
Format	:StatusType
Description	The StatusType argument of the Status event identifies the type of status. 1 - Informational 2 - Instruction
Sample	

The only requirement for an IOP Module is that it substantially conforms to the interface described earlier. The architecture described in this section, and its associated sample source code, is designed to facilitate development of IOP Modules by third parties. It should be used as a guide for developing an IOP Module—it is not the only way an IOP Module can be designed.

The internal architecture described below is independent from the interface requirements for an IOP. The IOP interface must be implemented regardless of whether or not the IOP is designed with the architecture described in this section. The basic IOP architecture is illustrated in FIG. **48**.

The IOP Module has a fixed set of features that it needs to perform:

Interface with the Server Module

Execute its operation when its Go( ) method is called

Respond to requests by the Server Module to configure its settings

Although any IOP Module that meets the IOP API requirements specified earlier will function properly, the proposed architecture simplifies the development of IOP Modules and ensures greater flexibility.

The internal Interface class has two purposes:

Communicate with the Server Module

Marshall requests to, from, and between the Execute and Configuration classes

In order to communication with the Server Module the Interface class must support the COM protocol. All modules within VC communicate via COM. This class should be created with the exact API specified earlier. Additionally, the Interface class should maintain the Feedback object passed in by the Server Module's Go method. This way all communication to the Feedback object will be handled by the Interface class, rather than by the Execute or Configuration classes.

The primary purpose of the Execute class is to execute the Go method when it is called by the Server Module. This is the core functionality of the IOP Module. Each IOP Module will have its own mechanism for executing its part of a virtual copy operation.

Any configuration information is assumed to have been passed to the Execute class by the time it is being called. Since the Execute class does not directly communicate with the Configuration class, any information that needs to be shared between the two classes must be coordinated by the Interface class.

The Configuration class maintains all the configuration data necessary for the IOP Module to operate. This includes responding to the Server Module to:

Prompt the user with a Configuration dialog

Save the current configuration information to persistent storage

Restoring the last saved configuration information from persistent storage

Since the IOP Module is entirely responsible for these activities, any programming method that accomplishes these tasks is legitimate.

The many features and advantages of the invention are apparent from the detailed specification, and thus, it is intended by the appended claims to cover all such features and advantages of the invention which fall within the true spirit and scope of the invention. Further, since numerous modifications and variations will readily occur to those skilled in the art, it is not desired to limit the invention to the exact construction and operation illustrated and described, and accordingly, all suitable modifications and equivalents may be resorted to, falling within the scope of the invention.

For example, while the above discussion has separated the various functions into separate layers of functionality, the layers may be combined, physically and/or logically, and various functions may be combined together. While combining various functions into same or common layers may make implementation details more cumbersome, nevertheless, the functions described herein may still be accomplished to advantageously provide some or all of the benefits of the invention described herein.

Further, as indicated herein, the present invention may be used to automate and/or manually expedite the migration of a program specific Application Programmer Interface from an original state into a generic interface by building an object for each engine. The object advantageously provides substantially uniform access to the engine and engine settings associated with the engine. The present invention amy be applied across a broad range of programming languages that utilize similar concepts as described herein.

I claim:

1. A computer data management system including at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet, comprising:

at least one memory storing a plurality of interface protocols for interfacing and communicating;

at least one processor responsively connectable to said at least one memory, and implementing the plurality of interface protocols as a software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications, wherein said software application comprises at least one of:

at least one input module managing data comprising at least one of paper and electronic paper input to the computer data management system, and managing at least one imaging device to input the data through at least one of a scanner and a digital copier, and managing the electronic paper from at least one third-party software applications; and

at least one module communicable with said at least one input, output, client, and process modules and external applications, and capable of dynamically combining the external applications with at least one of digital capturing devices and digital imaging devices.

2. A computer data management system according to claim **1**, wherein the one or more of the external devices and applications include a printer, a facsimile, and a scanner.

3. A computer data management system according to claim **1**, wherein the computer data management system includes the capability to integrate an image using software so that the image gets seamlessly replicated and transmitted to at least one of other devices and applications, and via the Internet.

4. A computer data management system according to claim **1**, wherein the computer data management system

includes the capability to integrate the electronic images into a destination application without the need to modify the destination application.

5. A computer data management system according to claim **1**, wherein the computer data management system includes an interface that enables copying images between physical devices, applications, and the Internet using a single "GO" operation.

6. A computer data management system according to claim **1**, wherein the computer data management system includes the capability of adding at least one of electronic document and paper processing with a single programming step.

7. A computer data management system according to claim **1**, wherein the software application comprises:

at least one output module managing the data output from the computer data management system, managing at least one imaging device to output the data to at least one of a standard Windows printer, an image printer, and a digital copier, and managing the output of the data to the third-party software application;

at least one process module applying at least one data processing to the data comprising the at least one of the paper and the electronic paper as it is being copied, applying additional functionality including at least one of workflow and processing functionality to the data comprising the at least one of paper and electronic paper as it is being copied, and applying multiple processes to a single virtual copy; and

at least one client module presenting the data comprising the at least one of paper and electronic paper as it is being copied, and information related to at least one of the input and output functions.

8. A computer data management system according to claim **1**, wherein the one or more of the external devices and applications integrates the computer data management system into an external application via one of running the computer data management system, as an external service and embedding the computer data management system as an embedded service.

9. A computer data management system according to claim **7**, wherein the server module includes:

enable virtual copy operation means for initiating, canceling, and resetting said computer data management system;

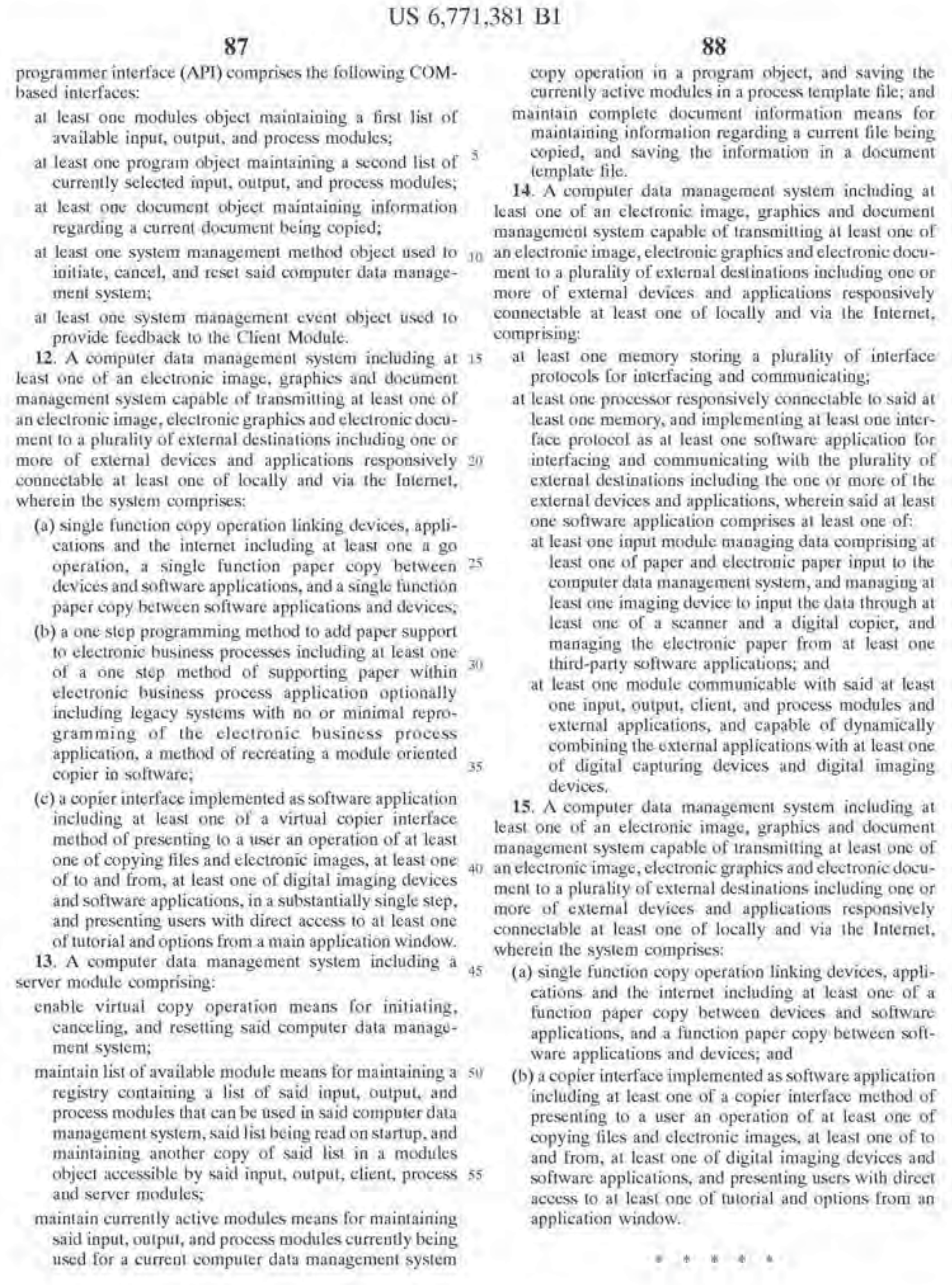
maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules;

maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer data management system copy operation in a program object, and saving the currently active modules in a process template file; and

maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

10. A computer data management system according to claim **7**, wherein the server module includes at least one server module application programmer interface (API).

11. A computer data management system according to claim **10**, wherein the at least one server module application



programmer interface (API) comprises the following COM-based interfaces:

at least one modules object maintaining a first list of available input, output, and process modules;

at least one program object maintaining a second list of currently selected input, output, and process modules;

at least one document object maintaining information regarding a current document being copied;

at least one system management method object used to initiate, cancel, and reset said computer data management system;

at least one system management event object used to provide feedback to the Client Module.

12. A computer data management system including at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet, wherein the system comprises:

(a) single function copy operation linking devices, applications and the internet including at least one a go operation, a single function paper copy between devices and software applications, and a single function paper copy between software applications and devices;

(b) a one step programming method to add paper support to electronic business processes including at least one of a one step method of supporting paper within electronic business process application optionally including legacy systems with no or minimal reprogramming of the electronic business process application, a method of recreating a module oriented copier in software;

(c) a copier interface implemented as software application including at least one of a virtual copier interface method of presenting to a user an operation of at least one of copying files and electronic images, at least one of to and from, at least one of digital imaging devices and software applications, in a substantially single step, and presenting users with direct access to at least one of tutorial and options from a main application window.

13. A computer data management system including a server module comprising:

enable virtual copy operation means for initiating, canceling, and resetting said computer data management system;

maintain list of available module means for maintaining a registry containing a list of said input, output, and process modules that can be used in said computer data management system, said list being read on startup, and maintaining another copy of said list in a modules object accessible by said input, output, client, process and server modules;

maintain currently active modules means for maintaining said input, output, and process modules currently being used for a current computer data management system copy operation in a program object, and saving the currently active modules in a process template file; and

maintain complete document information means for maintaining information regarding a current file being copied, and saving the information in a document template file.

14. A computer data management system including at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet, comprising:

at least one memory storing a plurality of interface protocols for interfacing and communicating;

at least one processor responsively connectable to said at least one memory, and implementing at least one interface protocol as at least one software application for interfacing and communicating with the plurality of external destinations including the one or more of the external devices and applications, wherein said at least one software application comprises at least one of:

at least one input module managing data comprising at least one of paper and electronic paper input to the computer data management system, and managing at least one imaging device to input the data through at least one of a scanner and a digital copier, and managing the electronic paper from at least one third-party software applications; and

at least one module communicable with said at least one input, output, client, and process modules and external applications, and capable of dynamically combining the external applications with at least one of digital capturing devices and digital imaging devices.

15. A computer data management system including at least one of an electronic image, graphics and document management system capable of transmitting at least one of an electronic image, electronic graphics and electronic document to a plurality of external destinations including one or more of external devices and applications responsively connectable at least one of locally and via the Internet, wherein the system comprises:

(a) single function copy operation linking devices, applications and the internet including at least one of a function paper copy between devices and software applications, and a function paper copy between software applications and devices; and

(b) a copier interface implemented as software application including at least one of a copier interface method of presenting to a user an operation of at least one of copying files and electronic images, at least one of to and from, at least one of digital imaging devices and software applications, and presenting users with direct access to at least one of tutorial and options from an application window.

* * * * *